无线传感器/执行器网络数据汇集及任务协作算法研究

Load-balanced Data Gathering and Task Coordination Algorithms for Wireless Sensor and Actor Network

易　军　唐云建　李太福　著

電子工業出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书系统地阐述了无线传感器/执行器网络的研究前沿问题——数据汇集及任务协作技术。全书共 9 章：第 1 章介绍无线传感器/执行器网络的特点和面临的挑战；第 2～5 章针对无线传感器/执行器网络的节点通信开销、拥塞、能耗、碰撞等问题，给出了不同应用目标下的数据汇集算法；第 6～9 章围绕传感器节点/执行器节点（Sensor-Actor，SA）协作和执行器节点/执行器节点（Actor-Actor，AA）协作，探讨设计了不同的任务协作算法。

本书的适用对象为计算机科学、电子工程专业的科研人员和研究生，以及电信业的研究和开发人员。

图书在版编目（CIP）数据

无线传感器/执行器网络数据汇集及任务协作算法研究 / 易军，唐云建，李太福著. —北京：电子工业出版社，2014.1

ISBN 978-7-121-22270-2

Ⅰ. ①无… Ⅱ. ①易… ②唐… ③李… Ⅲ. ①无线电通信－传感器②执行器 Ⅳ. ①TP212 ②TH86

中国版本图书馆 CIP 数据核字（2013）第 321755 号

责任编辑：刘海艳
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：720×1000　1/16　印张：12.75　字数：243 千字
印　　次：2014 年 1 月第 1 次印刷
印　　数：2 000 册　　定价：45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

无线传感器/执行器网络是一种新型信息获取和处理技术。在无线传感器/执行器网络中，传感器节点需要与执行器节点进行大量的紧密协作，共同完成对环境的监测和对事件的处理，因此出现了一种新的网络现象：传感器节点-执行器节点（SA）协作和执行器节点-执行器节点（AA）协作。本书面向无线传感器/执行器网络所涉及的理论与技术，针对传统的无线传感器网络相关协议和算法无法完全满足无线传感器/执行器网络新的需求，以节点之间的数据汇集和协同合作为研究手段，以提高网络实时性和能耗、负载均衡为目的，构建不同应用场景的 SA 协作和 AA 协作模型，基于最优化理论、群体智能优化、图论等计算方法展开理论与算法研究，并提供了最新的研究成果。

读者对象：

计算机科学、电子工程专业的科研人员和研究生，以及电信业的研究和开发人员。

本书特色：

1：对象新颖：无线传感器/执行器网络是一种新型无线传感器网络，代表了该领域的最新发展，尤其在物联网技术蓬勃发展的今天，作为其支撑技术的发展趋势，必将受到广大研究技术人员的关注。

2：视角独特：本书从传感器节点与执行器节点协同合作的视角，针对几种典型的应用场合，构建不同的协作模型，提出一系列数据汇集和任务协作算法，具有一定研究特色。

3：手段先进：本书将最优化理论，特别是群体智能优化技术引入无线传感器/执行器网络研究领域，提出了基于蚁群算法的动态负载均衡数据汇集算法、结合基于自适应学习策略的多邻域搜索策略与微粒群优化算法的多目标任务分派算法等，代表了未来的发展方向。

本书共 9 章，按如下方式组织：

第 1 章 绪论。阐述了本书的研究背景和意义，介绍了无线传感器/执行器网络基本理论知识，提出了本书的研究内容及成果、课题的来源和组织结构。

第 2 章 分析了碰撞退避机制对构造数据汇集树的影响，洪泛过程中的消息碰撞问题，以及数据汇集树的瘫痪问题，提出了动态交叉退避算法（DOBW 算法），并对 LDGT-CD 算法进行了仿真分析。

第 3 章 针对无线传感器网络数据汇集应用，提出了一种基于拥塞控制的负载均衡数据汇集树生成算法（LDGT-SPT），并对算法的正确性和算法的性能分别进行了理论证明和仿真分析。

第 4 章 针对无线传感器网络数据汇集应用中，由于数据源产生数据的速率不一致，静态负载均衡算法不能达到负载均衡的问题，提出了一种基于蚁群优化的负载均衡数据汇集算法（LDG-ACO），并对算法的性能进行了仿真分析。

第 5 章 针对无线传感器网络移动执行器数据汇集应用，提出了一种支持移动执行器的负载均衡数据汇集算法（LDG-MS），并对算法的性能进行了仿真分析。对无线传感器/执行器网络协作算法进行了分类综述，提出了协作算法的优化目标。

第 6 章 面向无线传感器/执行器网络，提出一种基于 SA 协作的理想分簇模型，确定簇内 SA 协作通信的理想半径，以此为基础提出分簇算法（CASA），并对算法的性能进行仿真分析。

第 7 章 针对事件频发区域的执行器节点执行任务频繁，导致能耗过大，任务排队时间过长的问题，提出一种实时 AA 协作框架，并对算法的性能进行仿真分析。

第 8 章 针对 AA 协作中有工序限制的任务分派问题，研究并设计带能量约束的单目标任务分派算法，并对算法的性能进行仿真分析。

第 9 章 针对 AA 协作中有工序限制的任务分派问题，进一步提出一种多目标任务分派算法，并对算法的性能进行仿真分析。

本书由易军、唐云建、李太福著，其中易军负责第 6～9 章的编写并统编全书，唐云建负责第 2～5 章的编写，李太福负责第 1 章的编写。

本书出版获得以下基金项目资助：国家自然科学基金（51374268、51075418、61174015），重庆市自然科学基金计划重点项目（cstc2012jjB40006、cstc2013jjB40007），重庆市自然科学基金（cstc2012jjA90011），重庆市教委科学技术研究项目（KJ121410），重庆高校创新团队建设计划（KJTD 201324）。

由于作者水平有限，加之时间仓促，书中难免存在不足之处，敬请广大读者批评指正。

目　　录

第 1 章　概　　论

1.1　无线传感器/执行器网络

1.1.1　无线传感器/执行器网络概述

随着近年来微电子（microelectronics）和无线技术的发展，由尺寸小、低成本、能量有限和具有无线通信能力的无线传感器节点[1]构成的无线传感器网络（Wireless Sensor Networks，WSN）越来越受到重视。无线传感器网络由成百上千的传感器节点组成，用来监视多种物理变量，如温度、湿度、声音、压力、振动等[2~3]。无线传感器网络有着广泛的应用场景，包括环境监控、目标跟踪、自然灾害预测（如地震、森林火灾）、家庭智能、智能交通控制和军事场合[2~4]等。然而在一些复杂场景，为了支持特殊任务的执行，要求传感器节点与更高性能设备的协作。因此一种由无线传感器网络衍生而来的新型网络就应运而生，这就是无线传感器/执行器网络（Wireless Sensor and Actor Networks，WSAN）[5]。它的应用涵盖广泛，从温室或者建筑物的节能控制，到生化或者核子袭击的检测，从工业过程的自动化到系统的通风量和温度的控制等[6]。

无线传感器/执行器网络由大量的传感器节点和少量执行器节点构成。传感器节点负责从物理世界收集信息，而执行器节点负责根据信息做出独立的判断，并执行相应任务[7~10]。如图 1.1 所示，用户可以通过汇聚（执行器）节点远程监控和发出执行指令。在无线传感器网络中，传感器节点一旦部署完成，不再移动或者改变。而传感器节点本身的资源有限，延长网络的生存期成为无线传感器网络的主要目标。然而对于无线传感器/执行器网络来说，传感器节点一般固定，但在某些场合也具有移动性；执行器节点具有移动能力、更高的计算能力、更强的传输能力和更多的电池能量[8~11]。因此，无线传感器/执行器网络与无线传感器网络的相同点如下。

（1）网络规模大。无线传感器/执行器网络依靠大量的传感器节点采集信息，需要在指定区域内部署相当数量的传感器节点。在一些场合，如森林防火和环境检测，传感器节点分布范围还相当广，所以需要解决网络的健壮性和容错性问题。

（2）传感器节点资源有限。传感器节点作为一种微型嵌入式系统，体积微小，计算能力、存储能力十分有限，需要完成监测数据的采集和转换、数据的管理和处理、响应执行器节点的任务请求和节点控制等多种工作。因此，需要解决如何在有限计算能力的条件下进行分布式合作的信息处理问题。

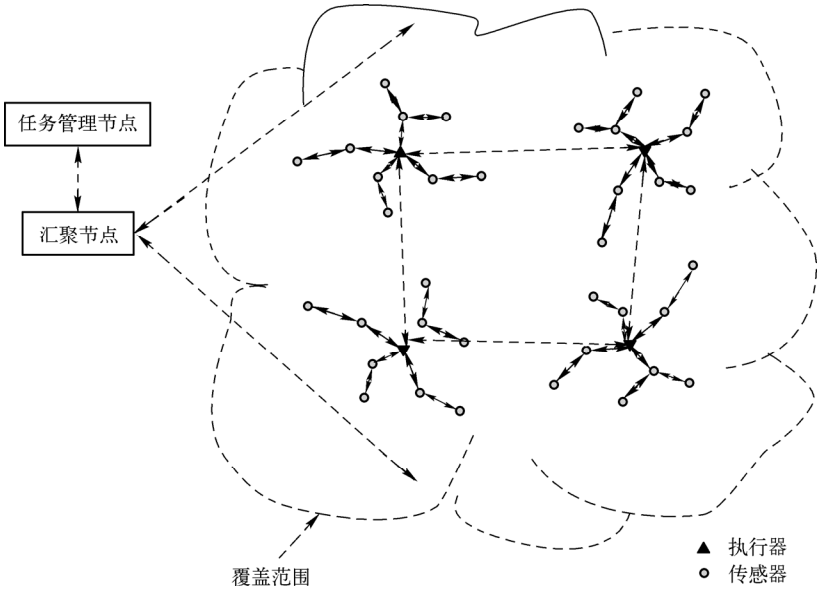


图 1.1　无线传感器/执行器网络物理结构图[8]

Fig. 1.1　The physical architecture of WSANs[8]

（3）节点能量有限。传感器节点由自身携带电池供电，大多部署在无人区或危险地带，无法更换电池；执行器节点相对传感器节点，虽然能量富裕得多，但是由于肩负执行任务，需要消耗更多的能量，因此同样存在能量有限问题。如何高效使用能量，达到网络能耗均衡，最大化网络生命周期是无线传感器/执行器网络面临的重要问题之一。

（4）自组织。传感器节点往往随机部署后，节点之间相互通信联结，形成网络。要求传感器节点具有自组织能力，能够自动进行网络配置和数据管理，通过拓扑控制机制和分层协议自动形成多跳无线网络系统。

（5）动态拓扑。无线传感器/执行器网络是一个动态网络，网络中节点的数目随时都在发生变化。节点可能因为电池能量耗尽或其他故障退出网络；也可以为了补偿失效节点增加监测精度而补充到网络中；执行器节点的移动同样会带来网络拓扑的动态变化。这都要求网络具有动态拓扑组织能力。

同时，由于少量异构节点——执行器节点的加入，与无线传感器网络相比，

无线传感器/执行器网络具有以下几个特征：

（1）实时性要求：一些应用场合要求网络对传感器数据进行快速响应。例如火灾应用中，网络必须对事件区域做出尽可能快的行动，事件的响应时间对控制事态进一步发展非常重要。此外，在执行过程中，传感器数据仍然可以有效地接收和发送，保持实时通信在无线传感器/执行器网络中也不可忽视。

（2）协作。在无线传感器网络中，通常由执行器节点充当数据收集和协作的协调中心。与无线传感器网络不同，在无线传感器/执行器网络中，出现了一种被称为传感器节点-执行器节点协作（sensor-actor coordination）和执行器节点-执行器节点协作（actor-actor coordination）的新的网络现象。传感器节点-执行器节点协作主要完成将事件特征从传感器节点向执行器节点传输功能。在接收到事件信息后，执行器节点需要和其他执行器节点协同合作，以便做出正确的判断并采取行动。

（3）异构性。无线传感器/执行器网络由传感器节点和执行器节点两种资源不同的设备构成，传感器节点间通信与执行器节点通信从方式和范围上都存在较大差异，因此需要不同的策略配合，才能克服异构带来的多样性和融合问题。

（4）移动性。在无线传感器网络中，一般考虑节点是静止的。但在无线传感器/执行器网络中，节点特别是执行器节点，接收到信息后需要移动到事发区域，执行相应的行动。

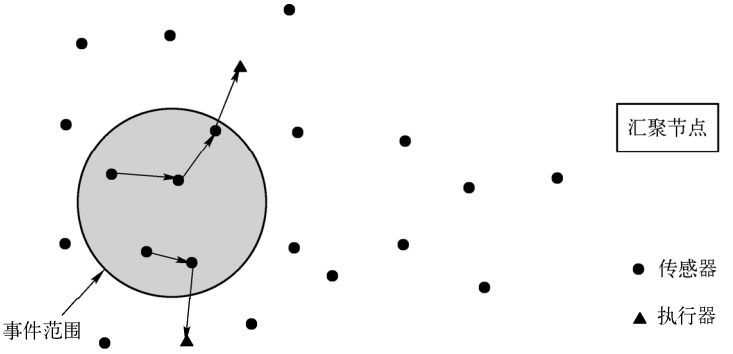
1.1.2 无线传感器/执行器网络体系结构及节点结构

1．无线传感器/执行器网络体系结构

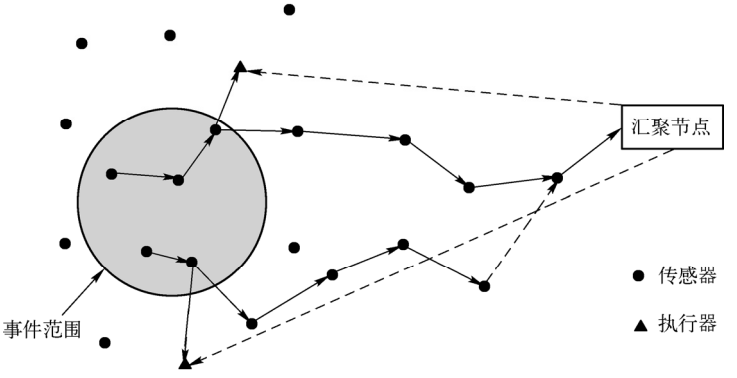
无线传感器/执行器网络是一个分布式系统，根据传感器节点采集数据和报告给执行器节点的方式，分为两种体系结构：自动（automated）和半自动（semi-automated），如图 1.2 所示。

在自动体系结构中，数据由传感器节点采集并直接发送到执行器节点，后者在不需要执行器节点的协调下直接与其他执行器节点协作采取行动。这种结构的响应速度非常快，适合对实时性能要求高的场合。在半自动体系结构中，传感器数据直接发送到一个中央控制器如执行器节点，由执行器节点处理数据并决定由哪些执行器节点承担任务。

无线传感器网络的通信过程主要发生在传感器节点与执行器节点之间，而无线传感器/执行器网络的通信则主要发生在传感器节点与执行器节点之间。



（a）自动体系结构



（b）半自动体系结构[8]

图 1.2　无线传感器/执行器网络结构

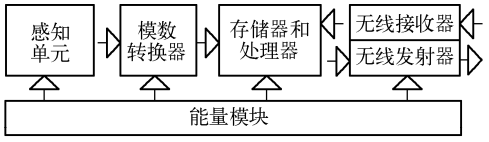
Fig. 1.2　Wireless sensor/actor network structure

2. 无线传感器/执行器节点结构

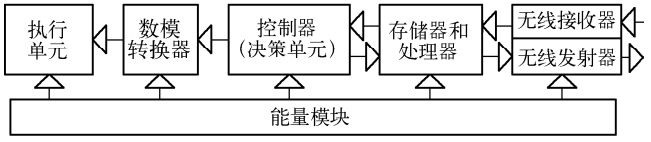
无线传感器/执行器网络由传感器节点和执行器节点两种异构设备组成，具有不同的性能和资源。

如图 1.3（a）所示，传感器节点由能量模块、通信子系统（无线接收器和发射器）、存储器和处理器、模数转换器（ADC）和感知单元组成。首先感知单元检测环境事件，然后数模转换器将模拟信号转化为数字信号，处理器再进行分析，并发送给附近的执行器节点。

图 1.3（b）所示的决策单元（控制器）读取由传感器输入的数据并输出执行命令。这些执行命令由数模转换器（DAC）转换为模拟信号，发送给执行单元执行。在一些应用中，传感器和执行器被整合到一个执行器节点上，这样的节点同时具有感知和执行能力，则图 1.3（b）的结构图中还应有感知单元和 ADC。



（a）传感器节点结构



（b）执行器节点结构[8]

图 1.3 无线传感器/执行器节点结构

Fig. 1.3 Wireless sensor/actor node structure[8]

机器人是一个集传感器和执行器于一体的好例子。如图 1.4（a）所示，由 Robotics Research Laboratories 设计的低空直升飞机平台可以完成地图绘制任务并可与自动移动机器人进行空地协作[12]，而且将来会具有更多的执行功能，如洒水、气体处理等。这使得无线传感器/执行器网络更加有效。图 1.4（b）是为军队设计的名为机器驴的自动战场机器人，由 Space and Naval Warfare Systems Command[13]和 Defense Advanced Research Projects Agency (DARPA)[14]研制成功。这些机器人能探测和标记地雷，运送武器，甚至在未来取代士兵进行作战。图 1.4（c）显示的 SKITs 是一组通过 UHF 频率以 4800 kb/s 的数据传输率进行通信的组网机器人[15]。这些机器人能够通过无线通信进行协作，执行由应用程序决定的任务。图 1.4（d）可能是世界上最小的智能机器人。

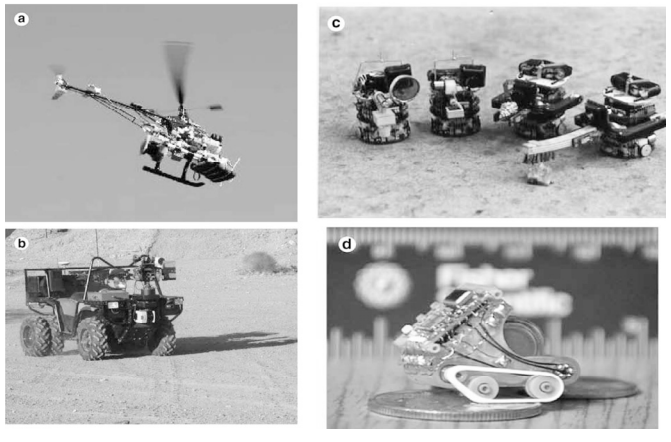


图 1.4 集传感器和执行器于一体的机器人例子[12~15]

Fig.1.4 Examples of robots as not only sensor but also actor[12~15]

1.1.3 无线传感器/执行器网络协议栈

无线传感器网络和无线传感器/执行器网络目前还不存在一个标准的协议栈。一般认为传感器节点和执行器节点协议栈基本由三个平面组成：通信平面、协作平面、管理平面[16]，如图 1.5 所示。

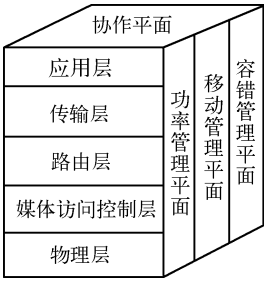


图 1.5 无线传感器/执行器网络协议栈[16]

Fig. 1.5 WSAN protocol stack[16]

1．管理平面（management plane）

管理平面的功能主要有三个方面：

（1）功率管理平面（power management plane）：管理功率如何使用。当节点的能量变低时，该平面通知协作平面，本节点不再参与感知、中转、执行等任务。

（2）移动管理平面（mobility management plane）：监视和注册节点的移动，以便保持网络的连通。

（3）容错管理平面（fault management plane）：负责节点出现故障的监控和解决。当感知单元或者执行单元失效，该平面立即把这一情况报告给协作平面。

2．协作平面（coordination plane）

协作平面负责根据从通信平面和管理平面接收到的数据进行判断，并做出决策。当事件发生后，传感器节点相互通信，并报告给协作平面进行决策。传感器节点之间的协作决定哪些低能量的节点不参与转发数据，哪些节点参与多跳路由和数据汇集。更重要的是执行器节点需要与其他节点协作完成适当的动作。当事件发生后，所有执行器节点的目的都是做出正确的反应，决定由哪些执行器节点参与处理事件。因此执行器节点必须具备发起协作和谈判的能力，这些必要的社交能力在协作平面内定义[17]。

3．通信平面（communication plane）

通信平面负责从协作平面接收命令，并根据命令利用通信协议建立与其他节点的链路关系，包括通信信道的建立、路由的选择等。

（1）物理层

物理层提供无线介质的接口，主要负责无线收发器的激活和休眠，能量检测，链路质量保证，空闲信道评估，信道选择和消息包的发送和接收等。此外，该层还负责位调制和解调的无线链路，以及发送器与接收器之间的同步[18]。

（2）传输层（transport layer）

在无线传感器/执行器网络中，不但要求传统的可靠性，还必须支持实时性要求。为 ad-hoc 网络和无线传感器网络开发的传输层协议已经出现[18~20]。由于传感器节点-执行器节点的通信与执行器节点-执行器节点的通信是同时连续出现的，一个统一的传输层协议需要在两种情况下都能良好工作。然而，到目前为止，针对无线传感器/执行器网络，还没有一个将可靠性和实时性兼顾的传输层协议。

（3）路由层（routing layer）

在无线传感器/执行器网络中，路由层的主要挑战是按照能量有效性的原则，为源传感器节点选择一个执行器节点。当数据通过中转节点向执行器节点传输时，需要进行汇集处理以便获得高有效性，路由协议应当支持实时通信，兼顾到不同的优先级，保证在时限内送达到执行器节点[21,22]。

（4）媒体访问控制层（MAC layer）

媒体访问控制层的主要功能是负责事件信息从大量传感器节点向执行器节点的有效传输。此外，在某些应用场合，执行器节点可能移动，保持传感器节点与移动执行器节点的连通也是其另一功能。另外，信息的及时监测、发布和处理也是该层的应用要求[23~25]。

（5）应用层（application layer）

负责网络载荷的管理，并将携带一定信息的数据按照某种可理解格式或发送要求进行转换，以便为不同的应用提供服务。该层根据不同领域不同应用，可为军事、医疗、环境、农业等领域提供多种特定的服务[24,25]。

（6）跨层设计

目前的无线传感器网络和无线传感器/执行器网络大多是按分层思想设计的。然而这种设计由于不够灵活和不够优化导致性能低下，为了获得低能耗和低时延，跨层设计显示出比分层设计的优越性。在无线传感器/执行器网络中，一个导致事件监控低可靠性的因素就是网络拥塞。当传输层通过降低传感器节点的传输速率进行重新发送时，MAC 层也通过指数回退方式重新传输，由于两层独立而重复的功能设计会导致效率低下。通过跨层设计，每层协议共同分享数据，效率明显提高。跨层设计还可以用于数据包的大小优化上，还广泛应用于功率控制、QoS 性能保证上[19,20,22]。

1.1.4 无线传感器/执行器网络的应用领域

无线传感器/执行器网络作为无线传感器网络的重要延伸，继承了无线传感器网络随机布设、自组织、自适应的特点，可广泛应用于军事、环境、医疗、家庭和办公等领域；同时增加了执行器节点，使得网络功能更加强大，为下一代网络——物联网提供了支撑技术。基于传感与/或执行能力的节点，并结合计算与通信功能，无线传感器/执行器网络可以实现各种类型的应用。

1．环境监测

传感器节点的低成本、易布设使得人们对室内外大面积进行监控成为可能，将网络化传感器节点部署在灾害事故频发区域，及时对物理结构变化做出反应，为人们预防与决策提供重要的数据参考[26~28]。环境监测方面的应用场合包括鸟类、动物、昆虫的迁徙跟踪，大面积地质结构监控[29]，精准农业，森林火灾、水灾、化学污染的预报检测等[30~35]。

2．药品管理与卫生保健[36~41]

无线传感器/执行器网络在医疗方面的应用：

（1）对人类生理数据的无线监测。无线传感器网络可以长期地收集人类的生理数据，供给医学研究应用。安装在身体上的传感器节点可以监控人类的行为，使医生尽早地发现病症。

（2）在医院对医护人员和患者进行追踪和监控。可以让每个患者佩戴可以完成特殊任务的微型传感器。这些传感器可以实时监测患者的心率、血压等生命指标。医生也可以佩戴微型传感器来保持与患者和其他医护人员的联络。

（3）医院的药品管理。由于每个患者佩戴了标志病症和治疗药物的传感器节点，与药品上的电子标签对照一致，可以大大降低给患者误用药物的概率。

3．军事应用[42]

由于无线传感器网络具有密集型、随机分布的特点，使其非常适用于恶劣的战场环境中，包括监控兵力、装备和物资，目标追踪，战争损伤评估，核、生物和化学攻击的探测与侦察等。

（1）用于跟踪敌人的军事行动，收集、处理、发射敌方信息。美国 Dust 公司研制的智能微尘（smart dust）[43]是一个具有计算机功能的超微型传感器，它由微处理器、无线电收发装置和使它们能够组成一个无线网络的软件共同组成。将一些微尘散放在一定范围内，它们就能够相互定位、收集数据并向基站传递信息。

（2）用于目标定位。目标定位网络嵌入式系统技术（Network Embed System Technology）是美国国防高级研究计划局主导的一个项目，它将实现系统和信

息处理融合。2003 年该项目成功验证了能够准确定位敌方狙击手的传感器网络技术，它采用多个廉价音频传感协同定位敌方射手并标志在所有参战人员的个人计算机中，三维空间的定位精度可达到 1.5m，定位延迟仅 2s，甚至能显示出敌方射手采用跪姿和站姿射击的差异。

（3）防核生化袭击。美国 Cyrano Sciences[44]公司已将化学剂检测和数据解释组合到一种专有的芯片技术中，称为 Cyrano NoseChip。基于这一技术可创建一个低成本的化学传感器系统，捕获和解释数据，并提供实时告警，以应付恐怖分子使用化学武器进行的攻击。

（4）战场环境侦察与监视。美国陆军最近确立了“战场环境侦察与监视系统”项目。该系统是一个智能化传感器网络，可以更为详尽、准确地探测到精确信息，如一些特殊地形地域的特种信息（登陆作战中敌方岸滩的翔实地理特征信息，丛林地带的地面坚硬度、干湿度）等，为更准确地制定战斗行动方案提供情报依据。该系统组由撒布型微传感器网络系统、机载和车载型侦察与探测设备等构成。

无线传感器/执行器网络用于军事场合的应用还有很多，直接推动了以网络技术为核心的新军事革命，诞生了网络中心战的思想和体系。传感器网络将会成为 C4ISRT（Command，Control Communication，Computing，Intelligence，Surveillance，Reconnaissance and Targeting）系统不可或缺的一部分。

4．空间探索

探索外部星球一直是人类梦寐以求的理想，借助航天器布撒的传感器网络节点实现对星球表面长时间的监测，应该是一种经济可行的方案。NASA 的 JPL（Jet Propulsion Laboratory）实验室研制的 Sensor Webs[45]就是为将来的火星探测进行技术准备的，已在佛罗里达航天中心周围的环境监测项目中进行测试和完善。

5．家庭及办公自动化[46~51]

（1）随着技术的进步，智能的传感器和执行器可以嵌入吸尘器、微波炉、冰箱等家用电器中。这些家电中的传感器节点可以彼此交互或通过 Internet 与外部网络交互，使用户可以方便地对家电进行远程监控和操作。

（2）智能办公是指在室内进行目标定位，如书、重要物品等的定位。将传感器放于物品中，用户就可以知道保存在什么地方。

6．物流与运输[52]

在物流与运输方面的应用中，可以在货物（如私人包裹）中安置简易的传感器，在运输过程中对目标货物进行简单跟踪或者对商店及库存货物进行方便的管理。这是实现物联网的关键技术。

7．其他商业应用[53]

自组织、微型化和对外部世界的感知能力是传感器网络的三大特点，这些特点决定了传感器网络在商业领域应该也会有不少的应用机会。例如，嵌入家具和家电中的传感器与执行机构组成的无线网络与 Internet 连接在一起将会为我们提供更加舒适、方便和具有人性化的智能家居环境；文献[54]中描述的城市车辆监测和跟踪系统中成功地应用了传感器网络；德国某研究机构正在利用传感器网络技术为足球裁判研制一套辅助系统，以降低足球比赛中越位和进球的误判率。此外，在灾难拯救、仓库管理、交互式博物馆、交互式玩具、工厂自动化生产线等众多领域，无线传感器网络都将会孕育出全新的设计和应用模式[55]。

1.1.5 无线传感器/执行器网络实验仿真平台

任何一项算法都需要对其性能进行分析与评估验证，目前主要的验证方式分为实验验证与仿真验证。

1．网络测试平台

通过传感器节点建立网络测试平台，可以在实际应用过程中验证测试网络的协议和算法，不仅比较全面地包含了影响网络状态的各个因素，而且也避免了因模型简化导致的理论误差。因此无线传感器网络的平台测试技术以及测试平台的搭建越来越被人们所关注。国内外对于测试平台搭建技术的研究还处于初始阶段，现有的一些如 MoteWorks[56]和 Kansei[57]等无线传感器网络平台，虽然支持网络测试的功能，但都局限于特定的节点和应用，测试内容也比较单一。国内外许多研究者都致力于开发通用的测试平台或者针对单一应用（如定位、通信协议）开发专用测试平台。如何形成系统化、标准化的测试评估体系以及如何使测试平台无论在网络规模及应用范围上具有更好的扩展性，将成为无线传感器网络平台测试技术研究的重点。

2．仿真平台

网络测试平台可以分析算法在小规模网络中的性能，却难以评估算法在大规模网络环境下的性能，特别是包含大量节点的大规模无线传感器网络，很难通过实验来实现（实际上，上百个节点的实验已经比较难以管理与实现）。为了实现无线传感器网络的仿真，研究人员设计开发了许多仿真平台（或在现有平台建立无线传感器网络模型），如 NS-2[58]、OPNET[59]、SensorSim[60]、EmStar[61]、OMNet++[62]、GloMoSim[63]、TOSSIM[64]等。这些仿真平台各有特点，目前大规模无线传感器网络的算法仿真验证大多是在这些平台上进行的。

1.2 数据汇集算法研究现状

1.2.1 数据汇集算法的设计目标

无线传感器/执行器网络数据汇集算法在无线传感器/执行器网络数据汇集应用中起着至关重要的作用，决定了无线传感器/执行器网络数据汇集应用的性能优劣，因此，其设计思路非常重要。一个良好的无线传感器/执行器网络数据汇集算法通常需要综合考虑以下设计目标。

1．简洁性

由于无线传感器网络规模大，节点通常一次性部署使用，因此要求节点低成本，导致节点资源严格受限，所以需要无线传感器/执行器网络数据汇集算法设计简洁，尽可能减少算法的存储和计算开销。

2．能量高效性

对于电池供电的无线传感器网络节点，利用有限的能量高效地完成数据汇集任务是无线传感器网络数据汇集算法设计的关键。

3．可靠性

无线传感器网络通常应用于煤矿瓦斯监控、核反应堆安全监控、工程爆破和军事防御等领域，这类应用要求无线传感器/执行器网络数据汇集算法能够非常可靠地运行。

4．自适应性

无线传感器/执行器网络通常工作于比较恶劣的环境，节点容易发生故障，甚至部分网络被破坏。部分旧节点失效后，可能会随时补充新节点。同时，在某些应用环境中，节点位置可能发生较频繁的移动，网络拓扑经常变化。因此，需要无线传感器/执行器网络数据汇集算法能够自适应网络环境的各种变化。

5．可扩展性

无线传感器/执行器网络的应用范围可大可小，比如，智能家居和社区安全的网络规模较小，森林防火和水系流域检测的网络规模则较大。因此，要求无线传感器/执行器网络数据汇集算法能够满足不同网络规模的应用和正在运行的网络的扩展需求。

6．服务质量保证（QoS）

服务质量是网络性能好坏的最终体现，用户总是希望得到数据丢包少、传输延时低（响应及时）的网络服务质量。因此，QoS 在无线传感器/执行器网络

的部分应用中要求很高。

7．安全性

对于要求信息安全的无线传感器/执行器网络数据汇集应用，需要无线传感器/执行器网络数据汇集算法具有信息认证、信息加密、入侵检测、防止敌意攻击（如 Sybil 攻击[65]和 DoS 攻击[66, 67]）等手段。

由于无线传感器/执行器网络的应用相关性，不同的应用环境对网络性能有不同的要求，因此，无线传感器/执行器网络数据汇集算法通常侧重于以上设计目标的部分内容。

1.2.2 数据汇集算法面临的挑战

无线传感器/执行器网络节点的固有特征，如节点能量供应、计算和存储能力有限，链路带宽有限等，使得无线传感器/执行器网络数据汇集算法的设计非常困难。加之应用环境恶劣、情况复杂多变，使得其数据汇集算法的设计极具挑战性，主要表现在以下几个方面。

1．开销问题

由于无线传感器网络节点的硬件资源非常有限，主要表现为计算和存储能力有限。因此，需要严格控制无线传感器网络数据汇集算法对硬件资源的开销，其中包括选择合理的网络结构、设计简洁的路由建立和维护算法等。

2．能耗问题

在无线传感器网络中，每个节点都可能扮演双重角色（数据采集器和路由器），而绝大部分情况下依靠电池供电，能量资源十分有限。通信部分在无线传感器网络节点能量消耗比例中占据主要部分。因此，为了节约能量，必须减少在数据汇集过程中不必要的通信开销，严格控制数据汇集算法在路由建立、路由维护过程中的额外能耗。这就要求数据汇集算法能够合理规划数据汇集的路径，减少因碰撞、拥塞等造成的丢包和重发，以及数据传输路径绕行时的不必要能耗。所以，能耗问题是无线传感器网络数据汇集算法重点考虑的因素之一。

3．拥塞问题

在节点大量部署的无线传感器网络应用环境中，每个节点都是数据源，数据产生的频率较高时，网络不可避免地会发生拥塞，怎样缓解网络拥塞是数据汇集算法面临的重要问题。拥塞通常在两种情况下产生：一种是由于节点的随机部署造成网络各个部分的节点密度不一致，在密度较高的区域往往数据产生量大、信道竞争激烈，造成频繁的丢包，拥塞情况显著；另一种是由于不合理的数据传输路径规划，导致数据过分集中在某些转发节点上，不但使得这些节

点的能耗加大，还造成在这些路径上严重的拥塞。拥塞不但会严重影响网络的服务质量，还会造成节点的能耗加剧，节点提前“死亡”，缩短网络寿命。所以，拥塞控制也是无线传感器网络数据汇集算法重点考虑的因素之一。

4．碰撞问题

由于无线电信道（绝大多数无线传感器网络采用的通信方式）的开放特性，隐藏终端、暴露终端问题不可避免。特别是在大规模网络环境中，节点通信半径较小的情况下，隐藏终端导致的碰撞问题非常明显。在实际的应用中，环境对无线电的衰减不一致导致通信半径不规则，碰撞问题更严重。碰撞不但会导致丢包，造成能量浪费，严重时还会影响路由工作。因此，无线传感器/执行器网络数据汇集算法的设计需要考虑碰撞避免，以及跨层设计等问题。

5．移动问题

在无线传感器网络的部分应用中，节点存在移动情况。节点移动包括三种场景：第一种是执行器节点固定不动，而其他节点在检测环境中移动，如野生动物生存情况监控、病人生理情况监控等；第二种是传感器节点固定不动，而执行器节点在检测环境中移动，如社区保安在监控区域巡逻、士兵在防御区巡逻等；第三种是所有节点都处于移动环境中，如海军通过安装有传感器的机器鱼群监测舰队周围的敌情，这时机器鱼群（传感器节点）和舰船（执行器节点）都处于不断的位置变化中。当前，PDA、3G 手机等手持设备的快速发展，第二种方式的移动数据汇集方式将会得到大量应用，因此本书针对该类型移动数据汇集方式做了尝试性研究。

1.2.3 典型的数据汇集算法

在众多的无线传感器/执行器网络数据汇集算法或协议（包括路由算法）中，存在一部分算法可以应用于数据汇集场景，本节仅讨论与数据汇集相关的典型算法或协议。在无线传感器/执行器网络的数据汇集算法中，网络的结构扮演了重要角色。基于不同网络结构的协议或算法有不同的适应环境，数据汇集的性能也不同。本书将无线传感器/执行器网络的网络结构分为平面、分簇和网格三种，并分别展开叙述。

1．基于平面网络的数据汇集算法

在平面网络中，除汇聚节点外的每个传感器节点都具有相同的通信能力和相同的电池电量供应，每个传感器节点在网络中的地位相同，形成完全意义上的对等网络，这给网络的部署带来了极大的方便。相对于其他网络结构，平面网络没有任何中心，个别节点的失效也不会对其他节点的数据传输造成严重的

影响，因此这种网络结构具有较强的鲁棒性。以下为几种典型的平面网络数据汇集算法或协议。

（1）洪泛（flooding）

洪泛是一种最简单的路由算法，它不要求维护网络的拓扑结构和计算相关路由，仅要求接收到信息的节点以广播方式转发数据包。例如，源节点希望发送一段数据给目标节点。源节点首先通过广播方式将数据副本传送给它的每个邻居节点，每个邻居节点再将数据以广播方式传送给周围其他节点，如此继续下去，直到数据传送到目标节点或者设定的生存期限（Time To Live，TTL）为止。如果生存期限足够长，可以使数据传遍整个网络，从而保证将数据传输给目的节点。在洪泛消息中加入了序列号等其他信息，可以使节点不重复广播已广播过的数据。在节点高速移动环境中，洪泛算法非常有效。但在通常的无线传感器网络应用环境中，洪泛算法无目的性的数据传输方式，会造成资源的严重浪费。

（2）定向扩散（Directed Diffusion，DD）[68]

DD 作为查询型数据收集的典型代表，其模型采用基于属性的命名机制来描述数据，并加入了数据融合和缓存机制。DD 在运行过程中包括路径建立、数据发送和路径增强三个阶段。路径建立阶段主要是通过汇聚节点以广播、多跳方式向网络中的所有节点发布兴趣（interest）消息，每个节点缓存接收到兴趣消息，建立从源节点到汇聚节点的梯度。数据发送阶段主要是源节点将采集到的匹配数据通过梯度路径发送到汇聚节点，中间结点通过本地化规则实现数据融合。而路径增强阶段主要是汇聚节点在收到数据后向数据到达最快的邻居节点发送增强消息，依此类推，建立一条从源节点到汇聚节点的主路径。DD 的工作流程如图 1.6 所示。

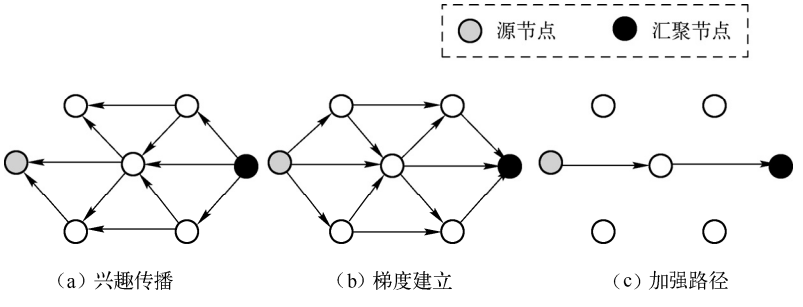


图 1.6　定向扩散机制

Fig 1.6　Scheme of direct diffusion

当被查询的数据源节点为多个或者全部传感器节点时，形成“多对一”的数

据流，数据收集就变为数据汇集。DD 引入了网络梯度的概念，可以找到从源节点到目标节点的最优路径，并使用了本地化算法，能够很好地满足无线传感器网络对鲁棒性和可扩展性的要求。但由于 DD 仅仅将传输延时和可靠性作为路径增强阶段考虑的因素，没有考虑节点的负载均衡，容易导致网络寿命的缩短。

（3）ACQUIRE（Active Query forwarding In sensoR nEtworks）算法[69]

ACQUIRE 作为一种查询型数据收集算法，它将整个无线传感器网络看作一个分布式数据库，该数据库可以执行比较复杂的查询条件，如“报告区域 X 最近 7 天的温度数据”，“是否当前温度超过了 35℃？”等。ACQUIRE 的原理为：执行器节点发出查询消息，每个传感器节点收到查询消息后，根据查询内容从自身预先缓存的数据中反馈部分结果，并转发查询消息给其他节点。如果在预先缓存的数据中没有足够新的数据，传感器节点就请求 d 跳的邻居节点获取信息。一旦查询完成任务，就将查询结果按照最短的路线返回给执行器节点。所以，ACQUIRE 算法通过允许多个节点发送响应信息，从而可以处理复杂的查询任务。ACQUIRE 算法通过调整参数 d 值，从而提供有效的查询。当 d 等于网络直径时，算法的查询行为类似于洪泛。然而，若 d 值太小，查询将不得不传输更多的跳数。不难看出，为查询消息选择下一跳的策略是 ACQUIRE 算法优劣的关键，这可以借鉴基于信息增益选择下一跳节点算法（CADR 和 IDSQ[70]）和谣传路由[71]的思想。如果网络中的多个或者全部节点满足 ACQUIRE 的查询条件，则数据收集变为数据汇集。同样，ACQUIRE 算法没有在负载均衡上做任何优化。

（4）GEAR（Geographic and Energy Aware Routing）路由[72]

采用洪泛方式将查询消息传播到整个网络的开销很大。而数据查询内容经常包含地理位置属性，因此 GEAR 路由算法利用能量感知和地理信息传输数据包到目标区域。其核心思想就是通过某一区域而不是整个网络，限制兴趣消息的扩散范围，以节省路由建立过程中的能耗。

GEAR 路由算法假设每个节点能够获取自身的位置和剩余能量，通过邻居发现知道邻居的位置和剩余能量。查询消息的传播包括两个阶段。首先，执行器节点发出查询消息，网络节点根据事件区域的地理位置将查询消息传输到区域内距执行器节点最近的节点，然后该节点再将查询消息传播到区域内的其他所有节点。监测数据沿查询消息的反向路径向执行器节点传输。

GEAR 路由算法用实际代价（learned cost）和估计代价（estimate cost）两种代价表示路径代价。当没有建立从执行器节点到事件区域的路径时，中间节点使用估计代价来决定下一跳节点。估计代价包括节点到事件区域的归一化距离以及节点的剩余能量两部分。当监测数据沿着查询路径反向传输时，附带计

算每跳节点到事件区域的时间能量消耗值，该值即为实际代价，主要用于调整路径，优化路由。

节点计算自身到事件区域的估计代价计算公式：

$$c(N,R)=\alpha d(N,R)+(1-\alpha)e(N) \qquad (1.1)$$

式中，$c(N,R)$为节点 N 到事件区域 R 的估计代价；$d(N,R)$为节点 N 到事件区域 R 的距离；$e(N)$为节点 N 的剩余能量 α 为比例参数。$d(N,R)$和 $e(N)$都是归一化的参数值。

GEAR 路由算法在路径的建立过程中采用了贪婪算法，如果节点的所有邻居节点到事情区域的路径代价都比自身大，则会陷入路由空洞（routing void）。GEAR 路由算法采用反向通知的方法避免路由空洞。如图 1.7 所示，节点 S 要将数据传输给节点 D，由于节点 B 距离节点 D 更近，因此选择下跳节点 B。而节点 B 发现其邻居节点的代价都比自身大（可以假设图中黑色节点故障或者不存在了），故选取邻居中代价最小的节点 A 作为下跳节点，并将节点 B 到节点 A 的代价增加到自身的代价上，同时通知节点 S。当节点 S 再转发查询消息到节点 D 时就会选择节点 A 而不是节点 B 作为下一跳了。

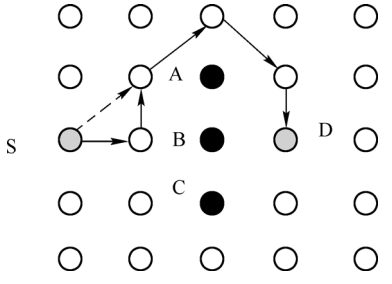


图 1.7　路由空洞

Fig. 1.7　Routing void

GEAR 路由算法定义估计代价为节点到事件区域的距离和节点剩余能量，并利用“捎带”机制获取实际代价进行数据传输的路径优化，从而形成能量高效的数据传输路径。GEAR 路由算法采用的贪婪算法是一种局部优化策略，适用于节点只知道局部拓扑信息的情况。其缺点是由于缺乏足够的拓扑信息，路由过程中可能遇到路由空洞，反而降低了路由效率。GEAR 路由算法中假设所有节点的位置固定不变或变化不频繁，适用于节点移动性不强的应用环境。

EAR 路由算法在事件区域内由于采用洪泛方式或者迭代方式转发查询消息，形成“多对一”的反向数据传输路径，而在事件区域外反向数据传输收敛到一条路径，容易造成该路径上的负载过重，能量快速消耗，这不符合负载均衡的要求。

2．基于分簇网络的数据汇集算法

分簇路由是一种通信效率高、扩展性较好的数据汇集手段。分簇网络数据汇集过程由一系列的簇首选择过程和数据收集过程组成。分簇网络数据汇集算法的核心在于分簇算法和调度算法等，而非多跳路由功能。在分簇结构中，根据功能不同，节点分为簇首节点和簇内节点。簇首节点用于管理簇内节点和收集簇内节点的数据。簇内节点主要负责采集环境数据，并接受簇首节点的调度。簇首节点与簇内节点之间采用单跳方式直接通信，簇首节点和簇首节点之间采用多跳方式通信，簇首节点最终与执行器节点相连。在网络中占据多数的簇内节点接受簇首节点的调度，因此可以在没有任务时进入深度睡眠状态，这对延长网络寿命有极大的好处。通常情况下，簇首节点还可以执行数据融合算法，以减少传输到执行器节点的冗余数据量，进一步有利于网络节能。由于簇首节点的任务量比簇内节点重，因此簇首需要经常更替以均衡网络节点的能量消耗。

（1）LEACH（Low Energy Adaptive Clustering Hierarchy）协议[73,74]

LEACH 协议作为无线传感器网络分簇网络数据收集的典型代表，采用一种分布式聚类技术。该协议的基本思想是：以循环的方式随机选择簇首节点，将整个网络的能量负载平均分配到每个传感器节点中，从而达到降低网络能源消耗、延长网络整体生存周期的目的。LEACH 协议采用基于 TDMA/CDMA 的 MAC 层机制来减少簇内和簇间的冲突，并在簇首节点进行数据融合，然后将融合后的数据发送给执行器节点以减少数据传输数量。W. Heinzelman 的仿真实验表明，LEACH 协议仅需要占总数 5%的节点充当簇首节点，并且与一般的平面多跳路由协议和静态分层算法相比，可以将网络生命周期延长 15%。

LEACH 协议在运行过程中不断地循环执行簇的重构过程，每个簇重构过程可以用回合（round）的概念来描述。每个回合可以分成两个阶段：簇的建立阶段（setup phase）和传输数据的稳定阶段（steady state phase）。为了节省资源开销，稳定阶段的持续时间要大于建立阶段的持续时间。簇的建立可分成 4 个过程：簇首的选择、簇首节点的广播、簇首节点的建立和调度机制的生成。簇首节点的选择过程为：每个节点选择一个从 0 到 1 之间的随机数，如果随机数小于某个阈值 $T(n)$，那么，该节点称为当前这个回合的簇首节点。阈值 $T(n)$的计算公式为：

$$T(n)=\begin{cases}\dfrac{k}{N-k[r \bmod (N/k)]}, & n\in G_r \\ 0, & n\in G_r\end{cases} \tag{1.2}$$

式中，N 为网络中传感器节点的总数；k 为一个回合网络中簇首节点的数目；r 为已完成的回合数，G_r 为在剩余的（$N/k-r$）个回合中为成为簇首节点的集合。

选定簇首后，簇首节点通过广播方式告知整个网络，所有非簇首节点收到该消息后，根据信号强度决定是否加入到相应的簇中，并发送消息，通知相应的簇首节点。最后，簇首节点采用 TDMA 方式为簇内中的每个节点分配时隙。

在 LEACH 协议传输数据的稳定阶段，每个节点周期性地占用信道传输数据，这与数据汇集应用的特点非常契合，因此非常适合连续监控的数据汇集应用。簇首节点的不断更替使得网络能量均衡消耗，体现了负载均衡的思想。但是，由于 LEACH 协议要求每个节点都能够与执行器节点通信，并且每个节点都具备支持不同 MAC 协议的计算能力，因此该协议不适合在大规模的无线传感器网络中应用。另外，该协议没有说明簇首节点怎样分布才能够较均匀地遍及整个网络，所以存在多个簇首节点集中于网络的某个区域，而其他区域没有任何簇首节点的情况。最后，LEACH 协议假定在最初的簇首选择回合中，所有节点都携带相同的能量，并且成为簇首的节点在每个回合中消耗的能量大致相同。因此，该协议不适合节点能量不均衡的网络。

（2）PEGASIS（Power-Efficient Gathering in Sensor Information Systems）协议[75]

PEGASIS 协议是在 LEACH 协议基础上改进的链式路由协议。其基本思想是：为了延长网络寿命，节点仅需与最近的邻居通信，节点轮流承担链首节点，并且轮流与执行器节点通信，当所有节点都与执行器节点通信后，所有节点再进行新一回合的轮流通信。由于这种机制使得能量消耗能够均匀地分布到每个节点上，因此延长了网络寿命。不同于 LEACH 协议的多簇结构，PEGASIS 协议采用链式结构，每次选择链中的一个节点作为链首向执行器节点传输数据。这种链式结构给人感觉像是平面网络结构，但由于每个节点都能够与执行器节点直接通信，相当于将多个簇首转化为一个簇首，因此本书仍将 PEGASIS 协议作为一种分簇网络协议来讨论。

PEGASIS 的链形成过程如图 1.8 所示：节点 C 被选为链首后，将链首标志向周围节点广播，收到链首标志的节点 A 将数据传输给节点 B，节点 B 融合节点 A 和自身产生的数据后，将数据传输给链首节点 C。同理，节点 E 将数据传输给节点 D，节点 D 融合节点 E 和自身数据后，将数据传输给链首节点 C。节点 C 融合节点 B、节点 D 和自身产生的数据，最后将数据传输给执行器节点。PEGASIS 协议中的每个节点运用信号强度测量与所有邻居节点的距离，并调整

信号强度以便仅一个邻居节点可以接收到信号，从而定位最邻近节点。那些彼此最接近的节点构成 PEGASIS 的链路，形成一条通往链首节点的路径。链路中的节点向靠近执行器节点方向的节点发送数据，节点收到数据后与自身数据进行融合处理。最后，由链首节点将数据传输给执行器节点。链路的构造按照贪婪方式进行。

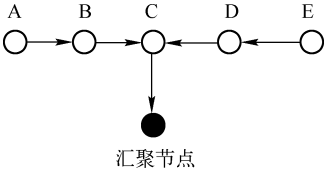


图 1.8 PEGASIS 的链式结构

Fig.1.8 Chain structure of PEGASIS

PEGASIS 协议的优点是减少了 LEACH 协议在簇重构过程中产生的开销，并且通过数据融合降低了数据传输量，从而降低了能量的消耗。仿真结果表明，与 LEACH 协议相比，PEGASIS 协议能够提高网络的生存周期近 2 倍。但是，PEGASIS 协议假设每个传感器节点能够直接与执行器节点通信，而在实际情况中，这样的假设很难成立。与 LEACH 协议一样，PEGASIS 协议假定所有的传感器节点都具有相同级别的能量，这不适合能量异构的网络情况。尽管 PEGASIS 协议避免了构建簇的开销，但协议要求节点动态地调整网络拓扑结构，这仍会带来较大程度的能量开销。最后，PEGASIS 协议的链式结构会使得距离链首远的节点产生较大的数据传输延时，且唯一的链首节点会成为网络瓶颈。

Lindsey 等人在 PEGASIS 协议的基础上进行扩展，提出了分层 PEGASIS 协议[76]，其目的是为了降低数据传输过程中的延时。为此，协议采取了数据并行传输的机制，并提出了两种方法来避免节点间的冲突和可能存在的信号干扰，即结合了信号编码方式的 CDMA 机制和空间分割的节点同时传输数据。基于 CDMA 的分层 PEGASIS 协议采用树状分层结构的方式，每一层选择的节点向更高一级的节点传输数据。该协议要求在每个回合的数据采集过程中，给定层的节点都向附近的邻居发送数据，所以接收数据的节点被提升为上一层节点。以此类推，顶层只有一个节点被保留下来并成为链首节点，最后形成一棵层次化的树状图，如图 1.9 所示。这种方法使得数据并行传输，大大减少了数据传输延时。仿真实验表明层次化的 PEGASIS 协议比原 PEGASIS 协议的性能更好。

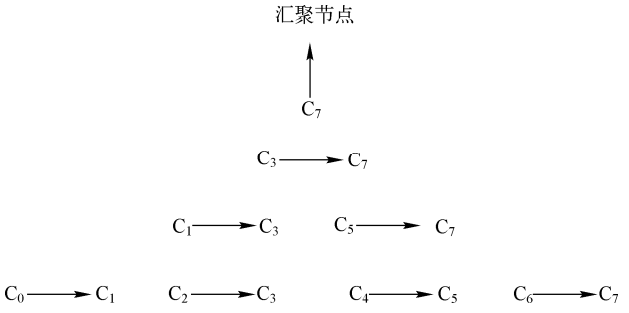


图 1.9　分层 PEGASIS 的数据收集机制

Fig.1.9　Data gathering scheme of PEGASIS

（3）TEEN（Threshold-sensitive Energy Efficient sensor Network protocol）协议[77]

TEEN 协议采用了与 LEACH 协议相同的多簇结构。为了减少数据的传输量，TEEN 协议簇首节点向簇内节点广播两个重要参数：硬阈值（Hard Threshold，HT）和软阈值（Soft Threshold，ST）。当节点首次采集到的数值大于 HT 时，便打开射频收发器传输数据，同时将此数值存入一个内部变量（Sensed Value，SV）中。节点再次采集到的数据如果大于 HT，并且与 SV 的差值大于或等于 ST 时，则发送数据，否则丢弃。节点每次发送数据后，用当前采集到的值更新 SV。因此，TEEN 协议的数据触发机制可以理解为：当数据处于用户感兴趣范围内，且有一定的变化时传输数据。较小的 ST 值可以较为精确地描述监测对象的变换，但会增加数据传输量和能耗，而较大的 ST 值，会使得监测精度不够。因此，ST 值的选择很关键。TEEN 协议的优点在于降低了数据传输数量，比 LEACH 协议节能，阈值可以动态改变，可以在准确性和节能性上做出平衡。TEEN 缺点在于，一旦阈值信息丢失，节点将不会工作，容易造成监测空白；如果监测对象变化不明显，或是节点停止数据传输，用户则无法判断节点是否出现故障。

针对 TEEN 协议的缺点，APTEEN（Adaptive Periodic Threshold-sensitive Energy Efficient sensor Network protocol）协议[78]对其进行了扩展。APTEEN 协议通过加入最大发送报告周期，用于强迫传感器节点向执行器节点传输数据，从而使得用户可以判断节点是否出现故障。

综上所述，分簇网络的簇首动态更替和簇首数据融合使得其具有能量有效性，但是分簇网络有其自身的缺陷。分簇网络需要节点具有功率控制功能和复杂的功率控制算法才能保证节能效果，分簇网络也不适合不能或无须进行数据融合的网络应用。分簇网络的功率控制策略，虽然减少了节点的能耗，但其节点的不对称通信半径，导致出现更多的隐藏终端。

3．基于网格网络的数据汇集算法

在分簇网络结构中，通常以簇的形式将大规模网络的数据汇集问题划分成小单位网络的数据汇集问题，属于分而治之的思想。如果节点能够知道自身的位置信息，就可以利用位置信息将网络划分成小单位，这样划分出来的网络单位的大小更加平均，这就是网格（grid）网络的思想。

（1）TTDD（Two-Tier Data Dissemination）算法[79]

TTDD 是一个典型的基于网格的数据收集算法，主要针对传感器节点静止，执行器节点移动，而且有多个执行器节点的应用场景。TTDD 算法的基本思想是：当多个传感器节点探测到事件时，选择一个节点作为发送数据的源节点，源节点以自身作为网格的一个交叉点构造格状网。源节点首先计算出四个相邻的交叉点位置，然后请求邻近交叉点位置上的节点成为转发节点，转发节点继续该过程直到网络的边缘。这样就形成一个如图 1.10 所示的网格。在网格构造过程中，转发节点记录了相关的时间信息和源节点信息。执行器节点在本地通过洪泛方式查询事件信息，当网格交叉位置节点收到查询消息后，将查询消息沿着网格交叉节点传输，最终使数据源节点收到查询消息。这种让查询消息仅在网格交叉点传输的思想大大减少了洪泛消息的数据量。根据 TTDD 算法的原理，可知该算法是一个事件类型的数据传输算法，本不属于数据汇集算法范畴，但是如果我们将构造网格的事件节点变成执行器节点，即以执行器为起始网格交叉点构造网格，传感器节点的数据通过网格交叉节点传输数据给执行器节点，就形成一种网格状“多对一”的数据流，因此从这个角度理解，可以将 TTDD 算法的思想作为一种数据汇集思想。

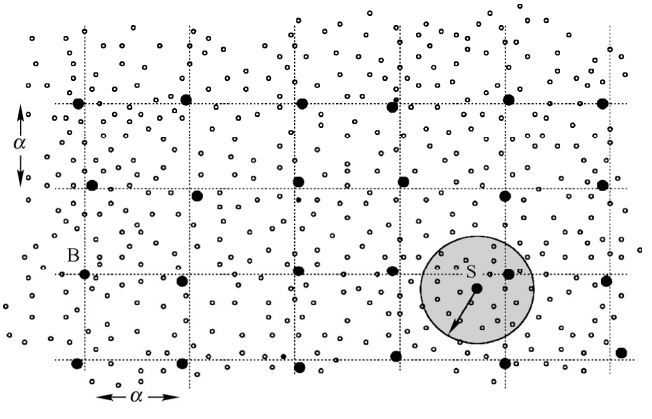


图 1.10　一个源节点 B 和一个执行器节点 S 的 TTDD 网格[79]

Fig. 1.10　One source B and one actor S of TTDD[79]

TTDD 算法需要解决怎样构造网格和怎样选取交叉节点。该算法假设网络中的每个节点都知道自身的位置，对于图 1.10 所示的网格构造发起节点 B，假设其坐标为（x, y），网格变长为 α，那么以 B 为起点的网格交叉点的坐标可以表示为

$$\{(x_i,\ y_i) \mid x_i=x+\alpha i,\ y_i=y+\alpha j;\ i,\ j=0,\ \pm1,\ \pm2,\ \ldots\} \tag{1.3}$$

TTDD 算法选择距交叉点最近的节点为网格节点，具体方法为：与交叉点之间的距离小于 $1/\alpha$ 的节点接收事件公告消息，否则不接收。收到消息的节点以广播方式发送自身的位置和与数据源信息，以此类推构成网格。转发事件公告时，以与交叉点的最短距离为标准，从而决定哪个节点作为最优转发节点。TTDD 算法的优点在于通过节点位置信息构建网格，执行器节点的查询消息只在网格的边缘传输，从而减少了网络洪泛消息，延长了网络寿命。但该算法也有一定的局限性：首先，构造网格的过程中没有考虑负载均衡，容易导致交叉点附近的节点，由于转发数据的负载过重而提前“死亡”，影响网络寿命；其次，该算法还要求网络节点的密度较大；最后，该算法没有对网格的大小进行优化，网格大小不好确定，而这个参数对网络的性能有很大影响。

（2）EEDD（Energy-Efficient Data-Dissemination）算法[80]

EEDD 算法针对 TTDD 算法中能量效率不高的问题，采用虚拟网格机制来减少网络能耗。初始化时，EEDD 的所有节点处于工作状态（working）。为了确定某个节点是否处于激活状态，节点在随机等待一小段时间后转化为探测状态（detecting），接着节点向周围所有邻居广播一个探测消息，其目的是为了探测在“节点感知区域”内的活动节点。如果在“节点感知区域内”某个处于工作状态（working）的节点的能量高于阈值 E_{gridhead}，则发送一个反馈消息。如果探测节点收到反馈消息的数目超过阈值 N_{node}，则认为自身为冗余节点，从而转入一个较长的睡眠期，否则节点进入工作状态（working）。阈值 N_{node} 取决于部署节点的冗余量和节点的感知范围。处于较长睡眠期的节点在经过 T_{sleep} 时间后又重新进入探测状态（detecting）。在节点从探测状态（detecting）进入工作状态（working）之前，需要发送一个网格头节点的声明消息。如图 1.11 所示，整个感知区域被划分为许多虚拟网格，R_{trans} 为传输半径，每个子网格又分为若干个调度时区，如图中所示的 P1、P2、P3、P4。

每个网格具有一个标志，记为 ID(a, b)，其计算方法为

$$\begin{cases} a=\dfrac{x-x_0}{\text{grid\_size}} \\ b=\dfrac{y-y_0}{\text{grid\_size}} \end{cases} \tag{1.4}$$

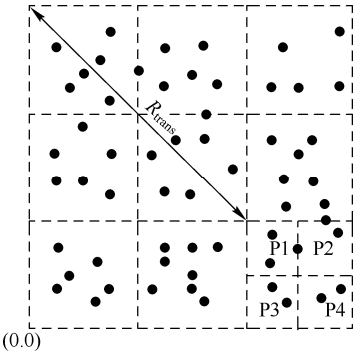


图 1.11 传感器节点的虚拟网格[80]

Fig 1.11 Virtual grids of sensor nodes[80]

式中，x_0、y_0 为系统初始化参数。为了保证所有的邻居节点能够直接通信，grid_size 设置为小于 $1/2\sqrt{2}*R_{\text{trans}}$，$R_{\text{trans}}$ 为传输半径。

每个节点在声明成为网格的头节点前都需要等待一个随机时间，其计算方法为

$$f(s)=\text{Random}(T_{\text{grid}})/\text{energy}(s) \tag{1.5}$$

式中，energy(s)为节点 s 的剩余能量。在等待过程中，一旦节点收到其他节点的声明消息就放弃本次尝试。由此可见，剩余能量越多的节点成为网格头节点的可能性更大，这有利于网络能量的均衡消耗，延长网络寿命。但是其探测活动节点的方法需要消耗较多的额外能耗，有进一步改进的必要。

网格网络的优点在于算法充分利用了节点的地理位置信息，能够合理划分网络区域和控制消息传输路径，缺点是要求节点密度较高，并目前获取节点地理位置信息的手段还不成熟或者成本还比较高。

1.2.4 负载均衡数据汇集算法研究现状

无线传感器网络数据汇集算法或协议通常是以节能为目标，最大限度地延长网络寿命。而在负载均衡数据汇集算法或协议中不仅要考虑怎样延长网络寿命，还要考虑怎样避免网络拥塞、降低关键节点的流量和均衡各个节点的数据转发量等其他因素。因此，本节主要介绍以负载均衡为目标的数据汇集算法研究现状。近几年来，国内外的研究人员提出了一些针对无线传感器网络的负载均衡策略。以下从单纯的拥塞控制、静止执行器情况下和移动执行器情况下的负载均衡数据汇集算法几个方面加以阐述。

1．拥塞控制机制

在拥塞控制方面，近些年来国内外的研究人员提出了一些针对无线传感器

网络的拥塞控制策略。Cheng Tien Ee 等人在文献[81]中提出一种拥塞控制机制 CODA。CODA 通过消息缓冲队列检测和周期性的信道负载检测来判断是否拥塞，当拥塞发生时启用开环控制策略，检测到拥塞的节点将压力消息（suppression message）向着数据源的方向反向传播，收到压力消息的节点根据本地情况判断是否继续传播压力消息，并采取一定的本地拥塞控制策略。CODA 使用开环逐跳拥塞反馈机制和闭环多源速率调节机制来缓解网络拥塞。在网络带宽利用率超过某个规定阈值时，触发闭环拥塞控制机制，通过 ACK 形式将发送速率控制信息反馈给各个相关节点。CODA 使距离执行器节点远的数据源节点获得传输的机会小于距离执行器节点近的数据源节点，这使得其公平性较差。文献[82]提出一种基于事件的拥塞控制机制 ESRT，在保证一定频度的传感器数据报文顺利到达执行器节点的前提下，尽量减少源节点的发包速度、节省通信能量。这种方法只适用于周期性报告的无线传感器网络应用，拥塞检测的周期比较长。文献[83]针对“多到一”树型拓扑的数据汇集应用，提出基于子节点数目的传输调度机制，保证每个节点发送相同流量的数据到执行器节点。根节点根据自身产生数据的速率和发送速率，为自身产生的数据建立一个队列，也为来自不同子节点的数据分别建立缓存队列，还同时记录每个子节点子树上的节点总数。子树上节点总数逐级统计，通过数据分组的方式“捎带”上传，数据产生速率通过控制命令或数据“捎带”下传给子节点。根节点根据每个子节点的子树上节点总数来确定每个缓存队列的发送概率。该文献还给出了根据子树节点数目按顺序和固定比例的分组调度算法，其主要目的是保证分组传输的公平性。文献[84]提出一种速率控制机制 IFRC，期望每个数据源至少可以获得拥塞的最小速率，使用 AIMD 控制方法，当发生网络拥塞时，所有相关节点先将自身的速率降低到拥塞节点的速率，然后各节点根据自身的状况慢慢提高，使全网的速率逐渐达到一个最佳分配。文献[85]提出根据区分服务的思想来减轻网络节点的内存和计算开销，通过指数级优先算法标定数据流等级，网络节点根据拥塞状态来动态调整报文过滤标准（节点门限），再根据所确定的报文过滤标准和报文优先等级采取相应的流量调节措施。该文献采用了队列溢出和队列等待时间进行拥塞检测，并通过控制报文来同步邻居节点间的过滤标准，保证了无线信道带宽分配上的公平性。文献[86]在分析了无线传感器网络特点的基础上，阐述了拥塞检测和拥塞避免的几种策略，重点介绍了基于速度控制、流量调度和传输调度等典型的拥塞解除算法，并对无线传感器网络拥塞控制技术的发展趋势进行了展望。

上述的拥塞控制机制大部分是基于点对点的速率控制，不适合数据汇

集应用。文献[83]虽然是针对“多到一”树型拓扑的数据汇集应用，但其主要目的是解决分组传输的公平性。文献[81～84]都是通过降低数据源的发送率来减少流量，以达到缓解拥塞的目的，其代价是牺牲了传感器的有效数据量。

2．静止执行器情况下的负载均衡

在执行器静止的情况下，张重庆等人在文献[65]中提出采用货币商品交易的思想，构建交易平衡的负载均衡网络，但在算法执行过程中需要保存大量的交易信息，存储开销比较大，并且在最坏情况下，遍历所有节点必须要遍历网络中的每一条边，因此算法的代价比较高。该算法仅适用于静态的网络结构，不能适应网络拓扑的变化。Pai-Hsiang Hsiao 等人在文献[87]中假设最初建立的路由树不是一个棵负载均衡的树，通过随机地选择邻居节点来对负载进行调整，因此该方法具有一定的盲目性。Hui Dai 等人在文献[88]中针对文献[87]的缺点提出利用节点的拓扑信息，采用拓扑控制来调整树的拓扑结构，以达到负载均衡的目的，但是对于资源非常有限的无线传感器网络节点，获取和保存全局的拓扑结构信息的方法是不可取的。Mark Perillo 等人在文献[63]中对无线传感器网络中“多对一”传输模式的负载不均衡问题进行了分析，将负载均衡问题转化为节点传输半径的优化问题，通过使用节点的位置信息，控制节点的传输距离以到达网络负载均衡。但该方法需要位置信息的支持，对于随机部署在复杂环境中的无线传感器网络节点获取位置信息代价比较大，并且该方法假定网络中所有节点都能与执行器节点或其他节点直接通信，这在很多应用场合都是不成立的。文献[89～95]采用基于分簇网络结构的负载均衡方法，但是分簇网络结构不但需要节点可以控制发送功率，还必须满足网络中的所有节点能与执行器节点或其他节点直接通信，这使得应用范围受到很大限制。文献[96～99]采用多路径路由策略来解决负载均衡问题，但这些策略都是基于点对点的数据传输思想，不适合“多对一”的无线传感器网络数据汇集应用。

3．移动执行器情况下负载均衡

在执行器移动的情况下，目前国内外学者提出了一些支持移动执行器节点的数据收发协议或算法，如 TTDD、SEAD[100]、CODE[101]以及文献[102, 103]中的算法。TTDD 算法通过事件节点的位置信息构建网格，执行器节点的查询消息只在网格的边缘传输，从而减少了网络广播消息，执行器节点在移动时通过指定代理节点来维持执行器节点移动过程中的路由路径。在数据源较多的时候，TTDD 算法为每个数据源构造网格的开销很大，当执行器节点移动出当前网格时，仍需要重新建立路由。SEAD 算法通过网格内部节点的休眠机制

来降低功耗，在执行器节点移动过程中通过扩展当前路由来维护路由路径，这样会造成严重的端到端的延时。CODE 算法虽然不用重新建立全局路径，但是需要其他路由协议的支持，增加了算法的复杂程度。TTDD、SEAD 和 CODE 都是基于查询或事件驱动数据传输模型设计，不适合数据汇集的应用。文献[111]提出一种局部路由维护机制，执行器节点周期性地发送执行器_Claim 消息，该消息用于邻居节点发现执行器节点和更替自身状态。收到执行器_Claim 的节点通过选举产生一个头节点，与执行器节点进行数据交互，一旦头节点失去了与执行器节点的联系就触发下一轮头节点的选举。该机制支持连续数据传输模型，但是为了保证路由的连续性，执行器节点必须以很高的速率发送执行器_Claim 消息，一方面会导致执行器节点快速耗电，另一方面在很大程度上降低了执行器节点的数据吞吐量。该机制没有提出一种可靠的选举方法，保证头节点选举过程中，不会出现因丢包而导致网络失去头节点，造成路由重建，从而产生较大的延时。并且，该机制需要其他路由手段支持（如 AODV），进一步增加了协议的复杂程度和节点存储器的开销。文献[103]提出一种适合数据汇集应用的移动执行器节点数据收发机制，该机制采用一个 SLM 节点来管理执行器节点的位置，所有发向执行器节点的数据先通过 SLM 节点获取执行器节点的位置，然后采用基于地理信息的路由进行数据传输。但是，如果 SLM 节点出现故障，就会立即影响到所有节点的数据传输，并且这种机制会造成 SLM 节点的负载很重，使得该节点的能量快速耗尽，严重影响网络寿命。

综上所述，负载均衡数据汇集算法通常采用传输调度、拓扑或功率控制、网络分簇、多经路由和构建负载均衡网络等策略或方法，这些方法各自具有一定优势，但也存在的一些缺陷和急待解决的问题，见表 1.1。因此，针对无线传感器网络的负载均衡数据汇集算法还有待进一步研究。

表 1.1　负载均衡策略/方法比较

Table 1.1　Compare of load-balance strategies

策略/方法	优　　势	存 在 问 题
传输调度	公平性好	负载均衡效果差
拓扑/功率控制	负载均衡较好	需要位置信息；获取和保存全局的拓扑；容易产生大量隐藏终端
网络分簇	易于数据融合、调度和休眠控制、节能效果好	不适合无须/不能使用数据融合的应用；需要复杂的功率控制算法才能保证效果
多路径路由	可靠性高	不适合“多对一”数据汇集应用
构建负载均衡网络	负载均衡较好	算法的计算和存储开销极大，并要求通信的可靠性极高

1.2.5 负载均衡数据汇集算法评价指标

本书将无线传感器网络的负载均衡数据汇集算法的评价指标分为三个方面：网络性能指标、负载均衡程度指标和能耗指标。

1．网络性能指标

网络性能指标是综合评价一个网络设计优劣的手段。网络性能指标主要包括吞吐量、丢包率和延时三个方面。

（1）吞吐量（throughput）

吞吐量是网络的一个重要性能指标，通常的测试是点对点的数据流量测试，即设置一对数据收发节点，在测试过载中不断提高发送节点的数据发送速率，同时观察接收节点收到数据包的情况。但在无线传感器网络数据汇集应用中，每个节点都是数据源，而目的节点通常只有一个（执行器节点）。因此，通常采用不断缩短传感器节点数据采集/发送周期（增大数据发送频率），统计单位时间内执行器节点收到有效数据包的方式来测试和度量。由于网络拥塞等原因会造成碰撞加剧，甚至丢包，导致吞吐量降低。因此，在同一个网络环境中，如果数据汇集算法的负载均衡程度较高，其吞吐量也会相应增加，反之亦然。

（2）丢包率（packet loss rate）

丢包率是指测试中丢失数据包数量占所发送数据包的比例，通常在吞吐量范围内测试，与数据包长度以及包发送频率相关。由于网络拥塞等原因会造成丢包，导致丢包率增加。因此，在同一个网络环境中，如果数据汇集算法的负载均衡程度较高，其丢包率也会相应降低，反之亦然。

（3）延时（latency）

延时是指数据包从数据发送端传输到接收端接所经历的时间，延时越小，网络性能越好。对于无线传感器网络数据汇集应用，数据总是从传感器节点传输到汇聚节点（执行器节点），延时定义为数据包从传感器节点传输到汇聚节点所花费的时间。由于网络拥塞等原因会造成重发，导致延时增加。因此，在同一个网络环境中，数据汇集算法的负载均衡程度较高，其延时也会相应降低，反之亦然。

2．负载均衡因子

对于一个层次结构的网络，假设其深度为 N，给定层次 i，$i<N$，从第 i+1 到第 N 层的所有节点，设总共有 M 个，则这 M 个节点产生的数据流，必然全部通过第 i 层。假设第 i 层上有 n 个节点，其负载分别为 $L_1, L_2, \ldots, L_n$，且 $L_1+L_2+\ldots+L_n$=C（大于零的常量）。为了均衡负载，应该让 $L_1, L_2, \ldots, L_n$ 之间的

差值最小化。根据Chebyshev Sum不等式定义：$a \in C^N$，$b \in C^N$，$a=\{a_1,a_2,...,a_n\}$，$b=\{b_1,b_2,...,b_n\}$，且 $a_1 \geqslant a_2 \geqslant ... \geqslant a_n$，$b_1 \geqslant b_2 \geqslant ... \geqslant b_n$，得

$$n\sum_{k=1}^{n} a_k b_k \geqslant \left(\sum_{k=1}^{n} a_k\right)\left(\sum_{k=1}^{n} b_k\right) \tag{1.6}$$

令 $a_i=b_i=L_i$，可得

$$\frac{(\sum_{i=1}^{n} L_i)^2}{n\sum_{i=1}^{n} L_i^2} \leqslant 1 \tag{1.7}$$

当 $L_1, L_2, \ldots, L_n$ 相等（负载均衡）时，不等式左边达到最大值。因此，定义负载均衡因子为

$$\theta = \frac{(\sum_{i=1}^{n} L_i)^2}{n\sum_{i=1}^{n} L_i^2} \tag{1.8}$$

式中，θ 称为均衡因子，表示负载均衡程度。θ 值越接近 1，说明负载越均衡。均衡因子的详细说明参见文献[97]。

3．能耗指标

无线传感器网络能耗指标通常用网络寿命来描述。在无线传感器网络中，网络寿命一般定义为：从网络开始运行到出现 k%个节点“死亡”的时间长度。在仿真环境中，节点因故障“死亡”的情况不会出现，节点的“死亡”是由其能量消耗殆尽引起的。为了便于统一衡量无线传感器网络的能耗性能，其网络寿命可以简化定义为：从网络开始运行到第一个节点“死亡”的时间长度。负载均衡技术可以缓解网络拥塞，降低网络冲突，减少消息重发，降低关键节点的能耗，延长网络寿命。因此，在同一个网络环境中，如果数据汇集算法的负载均衡程度较高，其网络寿命也会更长，反之亦然。

1.3 任务协作算法研究现状

无线传感器/执行器网络由大量传感器节点和少量执行器节点组成，这样的异构特点要求节点间进行大量协作，共同完成数据采集、处理、判断和执行任务。协作问题从层次上可划分为传感器节点/执行器节点（Sensor-Actor，SA）协作和执行器节点/执行器节点（Actor-Actor，AA）协作两种。

1．SA 协作

通常，SA 协作有三个主要议题：

（1）SA 通信的主要性能要求

传感器节点与执行器节点通信的主要要求之一就是低时延，在一些应用场合（如火灾），网络的实时性对快速反应并执行具有重要意义。与无线传感器类似，低能耗也是 SA 通信的重要性能指标。

（2）SA 通信的对象确定

无线传感器/执行器网络通信的另一个要求是需要确定向执行器节点报告事件的顺序。例如，两个传感器节点同时向一个执行器节点或者通信覆盖重叠区域内的一些执行器节点报告两种不同类型的事件，为了执行任务的正确性，事件必须顺序报告。这种传感器节点采集信息方式称为“有顺序发送”（ordered delivery）。另外需要考虑的是多个传感器节点报告同一个事件，这些信息可能同时到达某一确定的执行器节点，必须保证信息同步，以便执行器节点在事件区域内执行一次相应的任务。

（3）事件的跟踪报告

对于发生在不同地点的事件，传感器节点不应该将信息传给那些检测出事件的执行器节点，而是应该传给与发生事件地点最接近的执行器节点集。这就要求传感器节点必须能够跟踪事件，并利用这些信息确定将要报告的执行器节点集合。

2．AA 协作

在无线传感器/执行器网络中，执行器节点除了与传感器节点通信以外，还有可能与其他执行器节点通信，共同协作完成执行任务。对于 AA 协作，一般发生于以下场合：

（1）由于执行范围限制，或者能量不充裕，接收到传感器数据的执行器节点无法在事件区域内执行任务。

（2）一个执行器节点不足以完成相应任务，需要其他相邻执行器节点合作。

（3）如果多个执行器节点收到同一事件信息，这些节点应该协商并选出一个执行器节点完成任务。

（4）如果要求多个执行器节点覆盖整个事件区域，则必须保证这些区域是非重叠的或者是互斥（mutually exclusive）的，确保执行器节点在该区域的统一行动。

（5）如果多个执行器节点从多个传感器节点接收到同一事件信息，必须确保这些执行器节点在同一时间统一行动，同步执行任务。

（6）多个不同类型的事件同时发生时，必须通过执行器节点之间协商，进行任务分派；任务也可能被要求串行执行，这种约束称为有工序限制（ordered execution）任务。

（7）在一个执行器节点收到事件报告后，如果事件正在向其他执行器节点执行区域扩散，该执行器节点将传感器数据或者执行命令直接发送给这些节点，这样传感器节点就无须把事件报告给邻近执行器节点，代替由传感器节点来对事件进行跟踪。

AA 协作的主要议题是：选择合适的执行任务的执行器节点集合，即根据任务类型进行任务分派（task assignment）。AA 协作最重要的要求之一是最小化任务完成时间，事件才能得到执行器节点的及时响应。在一些如火灾的应用场合，实时性是非常重要的，常常能在事件发生初期遏制事态进一步扩大。在保证满足一定的时延要求下，选择多少执行器节点参与协商，最终决定哪些节点承担执行任务，才能使得各执行器节点能耗均衡，则是另一个重要问题。选择过程中，首先接到事件报告的执行器节点并不一定是“最好的”执行器节点，可能该节点的剩余能量不足以完成该任务，或者该任务超出了该节点的执行范围。如何利用协商机制，找出“最好的”执行器节点或者集合，也是 AA 协作需要考虑的问题。

1.3.1 协作算法性能评价指标

1．实时性

无线传感器/执行器网络的主要功能除了采集物理区域的数据信息外，还必须根据数据做出快速反应，执行相应的任务。及时采取行动往往能在刚刚监控到异常数据后就制止事态进一步恶化。因此，实时性是协作算法必须考虑的重要指标，在某些场合如森林防火，实时性是最重要的评价指标。

2．能耗均衡

网络节点的部署一般是随机布置或者网格部署，然而真实世界的事件发生地点却呈现帕累托分布（pareto distributions），即少数地点发生了网络大部分事件，出现所谓的热点区域（hot zone），该区域内的执行器节点需要执行更多的任务导致能耗过大，影响网络生存寿命。通过执行器之间的协作，使得全网能耗均衡，是协作算法重要的设计目标和评价指标。

3．可靠性

无线传感器/执行器网络往往工作于危险、人员不可达地带，容易受到周围环境或自身原因（如电池耗尽、物理损伤）的影响而发生失效问题，同时无线

通信常存在多径传播、衰减、非视距、冲突等问题。因此要求协作算法必须具有很强的可靠性。

4．自适应性

无线传感器/执行器网络是一种动态发展的网络，节点可能因电池耗尽退出网络，新的节点会补充加入网络，执行器节点的移动性也会改变网络的拓扑结构。因此无线传感器/执行器网络协作算法需要能够适应网络中的各种变化。

5．可扩展性

网络规模的扩展对协作算法各种性能指标会产生较大影响，同时不同的应用场合，无线通信的表现差异性较大，协作算法必须在网络规模扩展和应用场合扩展两种情况下都有好的表现。

6．事件发生频率

对于以事件驱动的网络，事件发生的频率和地点对协作是一个很大的考验。真实世界的事件发生地点分布并不均匀，不是理论上的随机分布，如何利用协作解决热点区域的事件发生频繁对网络性能的影响，以及事件发生频率对执行器节点部署的影响，都是需要算法考虑的。

7．数据类型

随着无线传感器网络专用化的深入发展，传输的数据类型越来越复杂，一般分为控制信息、简单的传感器数据如温湿度、声音、图像、视频等。不同的数据类型需要分配不同的带宽资源，对执行器节点协作的应用要求也不一样。

综上所述，无线传感器/执行器网络协作算法的性能评价指标，也是其设计和实现的优化目标。指标间相互制约，相互影响，由于无线传感器/执行器网络的应用相关性，不同的应用场合有不同的侧重指标，必须要在各项指标间进行综合考虑。

1.3.2 协作面临的研究挑战

1．SA 协作的分簇模型如何确定

当事件区域内传感器节点向邻近执行器节点报告时，存在两种情况：多执行器节点（Multi-Actor，MA）和单执行器节点（Single-Actor，SA）[8]，如图 1.12 所示。

在 MA 情况下，每个传感器节点可以独立确定向哪个执行器节点报告，但是由于传感器节点之间缺乏协作，可能导致过多的不必要的执行器节点参与事件处理，总的能量消耗增大。解决这个问题的有效办法是分簇技术，将网络划分为以执行器节点为簇头的若干个分簇区域，形成层次网络。当传感器节点监

测到事件发生，向本簇执行器节点报告，由该节点决定采取单独行动或者与其他执行器节点协作完成相应任务。那么，网络的分簇模型如何确定，需要考虑哪些因素成为 SA 协作面临的首要研究挑战。

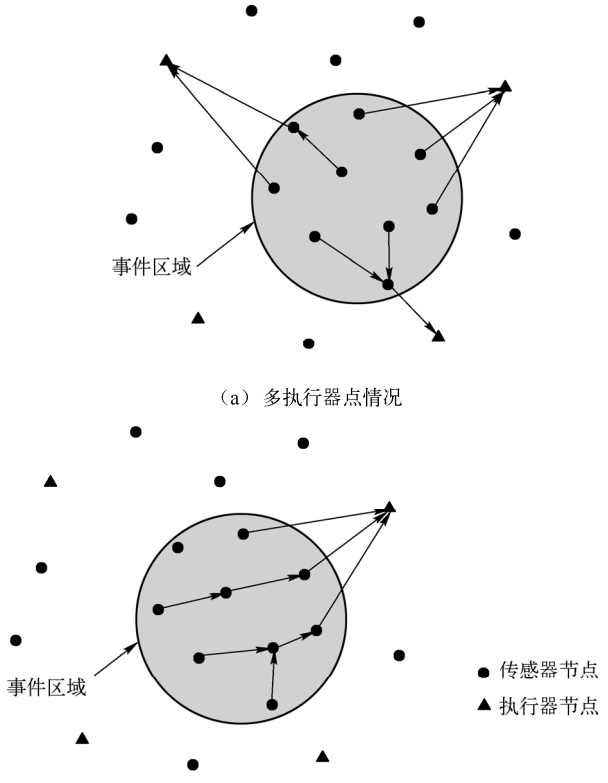


图 1.12 （a）多执行器节点情况和（b）单执行器节点情况[8]

Fig 1.12 (a) Multi-Actor (MA) vs. (b) Single-Actor (SA) [8]

2．簇内 SA 通信方式采用单跳还是多跳

对于簇内 SA 通信，存在两种方式：单跳或者多跳。传感器节点与执行器节点如果采用单跳通信，由于减少了中间转发时延，实时性相对较高，但是需要更大的传输功率；多跳方式则刚好相反。如何确定簇内传感器节点的通信半径是 SA 协作面临的另一个挑战。

3．AA 协作采用集中式还是分布式

目前，AA 协作方法主要分为集中式和分布式两种：集中式由某一个执行器节点统一决定最佳任务执行集合，所需信息多，计算量大；而分布式体系结

构相对于集中式结构，具有计算量小、通信量小、容错性强等优点，被广泛用于多执行器节点动态协作中。

4．AA 协作的复杂任务如何分派

一旦事件发生频繁，需要执行的任务数量增多，同时事件处理要求更高，任务越来越复杂，应利用执行器节点协作，动态调整候选节点范围，建立任务执行的代价评估机制，选出局部最优的执行器节点集合，使得完成任务时间最短，同时保证网络能耗均衡。

1.3.3 典型的协作方法

整个无线传感器/执行器网络是一个分布式系统，具有自组织特征。每个执行器节点都具有独立处理事务的能力，可以看成智能体，则无线传感器/执行器网络就是一种多智能体系统。基于多智能体的协作方法，同样可以借鉴应用于无线传感器/执行器网络中，主要的典型协作方法包括基于合同网及拍卖的方法、基于动态联盟的方法和组织结构设计法[96]。

1．基于合同网及拍卖的方法

Charles L.Ortiz 等人首次提出基于合同网和拍卖的系列方法，包括基于中心的任务分配方法、组合拍卖方法与仲裁方法[71]、迭代式组合拍卖方法[98,99]及动态仲裁方法[100]。基于中心的任务分配方法需要逻辑上的处理中心，后三种方法都是基于中心的任务分配方法的扩展算法。

（1）基于中心的任务分配方法

定义：M 代表一个四元组，$M=<A, T, u, P>$，其中 A 是 n 个智能体的集合，从中产生数据处理中心；T 是 m 个任务的集合；u 表示智能体执行任务的能力；P 是 T 或其子集的分配方案集合。通常设定一个全局目标函数 f，用于评价分配方案的优劣，其取值可根据所分配任务计算得出。

任务分配过程：首先扮演中心节点的智能体向其他智能体发出声明，包括任务分配方案和任务集合。每个智能体据此产生标的信息，与中心节点经过构造声明、声明发布和接收标的三个步骤的多次迭代，在满足终止条件或者接收到外部终止信号时终止，并由中心节点向智能体发送分配方案。

（2）组合拍卖方法与仲裁方法

在组合拍卖方法中，首先选出一个智能体作为拍卖者，根据事件产生一个任务集，然后将任务（包括已分配任务和待分配任务）发布给其他智能体。其他智能体收到消息后，根据自身执行能力从待分配任务中找出所有可以执行的任务。设这样的任务集合为 T，那么，该智能体有能力执行 T 的任意一个子集，设这样的子集的

集合为 $S(T)$；根据执行 $S(T)$中各个任务子集的效率对各个任务子集产生一个标的，标的表征该智能体执行对应任务集的能力；然后将各任务子集及对应的标的发送给拍卖者。随后拍卖者根据各个智能体发送过来的标的生成最优分配方案。

在仲裁方法中，从智能体中选出一个仲裁者，仲裁者负责对任务进行组合并产生分配方案，资源要求较大。仲裁过程仍然需要多次交互：仲裁者首先初始化任务分配方案 A（代表到目前为止发现的最好解），评价值为 m，随后搜索并产生新解 B，称为当前解。仲裁者把当前解广播给周围智能体，每个智能体对此做出响应，仲裁者对这些响应信息进行综合评价后获取一个评价值 n；如果 $n > m$，说明新方案 B 比旧方案 C 好，那么就以 B 取代 C。直到终止条件满足，当前方案为最优方案。仲裁方法可以用来处理实时性要求较弱的问题。

2．迭代式组合拍卖方法

迭代式组合拍卖算法将任务分配抽象成具有 m 个顶点的无向图，每个顶点代表一个任务，以概率 p 连接顶点对，p 表示任务间的相互作用概率。一个任务集是相互作用的，当且仅当由其对应顶点生成的子图是连通的。算法首先生成初始分配方案；然后根据方案初始化任务连接图，每个向量对应一个任务；最后按照两种情况进行迭代，改进分配方案；一是在两个非连接子图间增加一条边；二是根据新增加的边改进分配方案。在每一步的迭代过程中，针对当前新生成的连接图，组合拍卖会找到最优分配方案，当整个任务连接图是连通图时，即当增加一条边的时候，无法连接两个未连通的子图，迭代过程终止。

迭代式组合拍卖方法的优点是不需要标的的产生过程作为输入，智能体自行通过迭代找到最优解。

3．动态仲裁方法

动态仲裁算法的最大特点是通过建立并维护相互作用表，将一般的仲裁方法扩展成动态仲裁算法。相同之处在于：每个智能体在收到仲裁者发送的任务分配方案后，仍然针对分配的任务进行投标，再将标的发送给仲裁者。不同的是：智能体将不同类型任务之间的相互关系以相互作用表的形式存储在标的中。仲裁者通过迭代方式，利用相互作用表中的信息搜索任务分配方案。在每一次迭代过程中，仲裁者利用相互作用表进行更新，并构建概率分布表，表明各个智能体执行每类任务的可能性。仲裁者利用不断更新的相互作用表来调整智能体执行给定任务的可能性。

迭代式组合拍卖方法、动态仲裁方法可以用来处理实时性要求较强的问题。

4．基于动态联盟的方法[101,102]

Soh L.K 和 Tsatsoulis C.提出的动态联盟方法，主要思想是事件发生时，检

测到该事件的智能体成为发起者，根据任务和邻居智能体的执行能力，选择某些智能体作为初始联盟，发起者再通过与联盟成员进行多边一对一谈判，最后确定具体执行任务的成员，形成一个有效的联盟。该方法包括三个阶段：

（1）初始联盟阶段。发起者根据邻居智能体的潜在能力，选出一定数量的智能体形成初始联盟；再对任务进行初步分配，并根据该分配方案与联盟成员进行一对一协商，邀请能力强的参加组成联盟；收到邀请的智能体通过计算开销，决定加入还是拒绝；最终确定执行具体任务的成员。

（2）联盟终止阶段。当联盟形成后，所有协商终止，并随着任务的完成而解散。

（3）联盟学习阶段。为保证联盟形成的可靠性和快速，联盟的初始化和协商阶段都是采用基于案例的推理机制，所用案例是由学习过程获得的。

案例包括任务的描述，各智能体当时的位置、状态、能力；形成初始联盟的策略和协商策略；最终的结果。在案例推理过程中，先对问题进行描述并找出与该描述最接近的案例，参考其对应的策略加以改进使其适应目前各智能体状况。

5．结构设计法[103,104]

结构设计法由 Bryan Horling 等人提出，用于解决大规模协同问题。方法类似于分簇（cluster），即将目标区域划分为若干个矩形子区域，每个子区域有一个管理者，类似簇头（cluster-header），负责控制该区域的信息和数据。簇头还同时控制一个或者多个追踪器，协同本区域内的传感器节点追踪目标。传感器节点作为本区域成员，只与管理者和追踪器通信。不同的区域通过管理者通信。这样的划分减少了网络的通信量，便于进行大面积监控，并具有很好的扩展性。

综上所述，基于合同网和拍卖的方法可以较好地解决小型协作问题，并能满足实时性要求；而基于结构设计的方法适用于大规模协作，与无线传感器/执行器网络的特点相符合。然而，无线传感器/执行器网络中的协作以大量的传感器节点为基础，协作必须符合网络的结构特点。如果能首先采用结构设计法将网络划分为簇，簇内传感器节点与执行器节点通信，减少网络通信量，再由该簇执行器节点根据事件采用基于合同网和拍卖的方法与其他执行器节点形成动态联盟，一起协作有望解决大规模的任务分配问题。

1.3.4 典型的协作算法

1．DCF（Distributed Coordination Framework）算法[105]

DCF 算法是 Tommaso Melodia 等人提出的一种分布式协作框架。算法将事

件驱动分簇问题建模成整数线性规划问题，提出一个多状态分布式算法，每个传感器节点在空闲、启动、加速、聚合四个状态切换。源节点在发送事件数据时给数据包标上时间戳，以便相应的执行器节点计算该包到达的时延。事件可靠参数 r 表示已到达的数据包传输速度，并由执行器节点周期性广播出去。当 r 值低于所谓的事件可靠下限 r_{th}^{-}，传感器节点就必须通过加大发送功率，加速数据发送速度，减少中间转发时延；反之，当 r 值高于所谓的事件可靠上限 r_{th}^{+}，传感器节点减小发送功率以节约能耗。

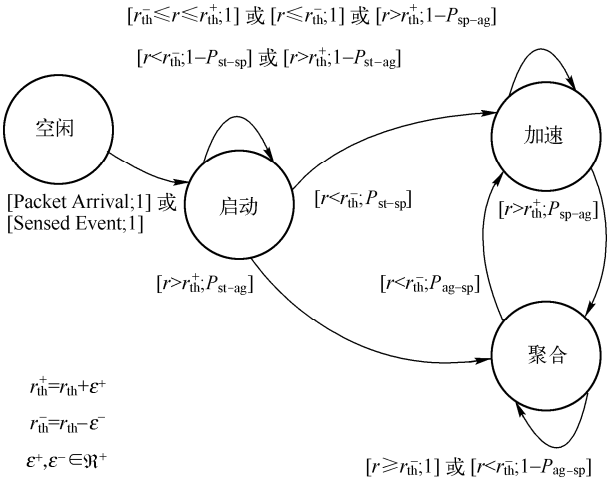


图 1.13　传感器节点的状态迁移图[105]

Figure 1.13　State transition diagram for a sensor node[105]

图 1.13 的状态迁移图说明了每个状态迁移的条件和概率，为 SA 协作提供了一种在时延和能耗之间平衡的自适应机制。但是由于事件可靠上限和下限无法精确计算，也没有给出当网络动态变化时该值的有关公式，使得本算法停留于设计阶段。

同时，算法将 AA 协作问题建模成混合非线性整数规划问题。在事发区域内，以达到任务的完成时限为约束，最大化执行器节点的平均剩余能量。算法只考虑了执行器节点协作中的能耗，需要从实时性和执行范围等多角度进行改进。该作者在文献[106]做的也是类似工作，不再赘述。

2．RCR（Real-time Coordination and Routing）算法[123]

Shah G.A.等人提出了实时协作与路由框架（RCR），适用于自动体系结构和半自动体系结构的 SA 协作通信。框架分为两个部分：DAWC 协议和 DEAR 协议。DAWC（Dynamic Weighted Clustering Algorithm，动态加权分簇算法）是一种用于动态分层配置分簇结构中的传感器节点，以能量有效为目的的分簇协

议。协议由簇的建立、簇头选举和邻居簇的发现三部分组成。带时延约束条件的能量感知路由算法（Delay-constrained Energy Aware Routing，DEAR）整合了向前跟踪和回溯两种路由技术，在给定时延 τ 内从源节点到执行器节点建立一条路由。考虑执行器节点的存在，在半自动体系结构下通过执行器节点，利用集中式 DEAR（C-DEAR）算法协同传感器节点与执行器节点工作；在没有执行器节点的自动体系结构下，利用分布式 DEAR（D-DEAR）算法完成 SA 协作。簇头在多条候选路径中选择一条最能量有效的发送数据，同时必须满足时延条件的限制。RCR 算法在分簇和在分簇结构下路由方面提供了新的思路，但是该算法没有对 AA 协作进行讨论，无法保证在任务分派阶段同样具有能量有效和实时性，从无线传感器/执行器网络的协作算法来看，内容不够完整。

3．RCF（Real-time Communication Framework）算法[124]

Edith C. H. Ngai 等人提出一种实时通信框架，分为两个阶段：首先在传感器节点向执行器节点传输数据时，提出一种实时的分布式算法，将监测区域划分为若干簇区域，同一簇区域内报告的事件数据只向外传输一组，避免多个传感器节点同时报告造成负载过大；同时最先发现事件的节点将数据按优先等级分为若干级，将最重要的数据先发送，提高实时性，再传送其他数据。该算法按照最小时延方式选择路由，并赋予更重要的数据以更高的优先级。当数据被执行器节点接收后，不需要等到收齐全部数据即开展协作，多个执行器节点将各自收到的地理信息进行组合，选出最合适的节点执行任务，被选中的执行器节点向周围节点报告移动后的地理位置，其他节点更新该点地理信息。然而该算法过分依赖于定位技术所测的地理位置，定位精度直接影响算法的有效性，当网络发生动态变化时，需要重新向邻居节点广播位置信息，以便重新定位，计算开销非常大。

4．CA（Communication Architecture for Mobile WSAN）算法[107]

Tommaso Melodia 等人在提出了静态无线传感器/执行器网络的协作框架之后，专门为移动执行器节点的协作和通信问题提出 CA 算法。算法首先引入了一种异构定位管理策略：当传感器节点利用上次接收到的位置更新信息，通过卡尔曼滤波（Kalman filtering）预测到执行器节点的移动，执行器节点即可在有限范围内广播位置更新信息。这种专门针对无线传感器/执行器网络的定位管理策略克服了以往同类算法的不足，传感器节点减少了 75%的位置更新所需能耗。另外，算法基于瑞利衰落信道影响下的地理路由提出一种跨层（路由层/物理层）机制，得出一个简单而优化的转发规则，从能耗的角度进行优化。在低、中级载荷情况下，利用功率控制技术在数据发送过程中控制时延；在高级载荷情况下，利用网络层的拥塞控制机制，促使多个执行器节点分担事发区域

的载荷量。算法最后引入一个 AA 协作的新模型，用于协调多个执行器节点的移动和动作，共同处理多个并发事件。特别是选取一个执行器节点集合组成一个团队，驱使该团队向事发区域移动并执行特殊任务。CA 算法的缺点是使用的技术如卡尔曼滤波要求实际系统的运动模型已知，如果事件发生后执行器节点的运动规律和各种干扰的统计特性不能准确预知的话，滤波精度就要降低，甚至造成滤波器的发散。同时卡尔曼滤波的计算量也不宜用于实时性比较高的场合。另外提出的拥塞控制需要计算拥塞因素、方向因素、距离因素，同样存在计算量大和实时性不高的缺点。

5．AAC（Actor to Actor Communication）算法[108]

Dimitris Vassis 等人详细分析了以 IEEE 802.11 协议为基础的单信道 AAC 策略和多信道 AAC 策略的性能差异。

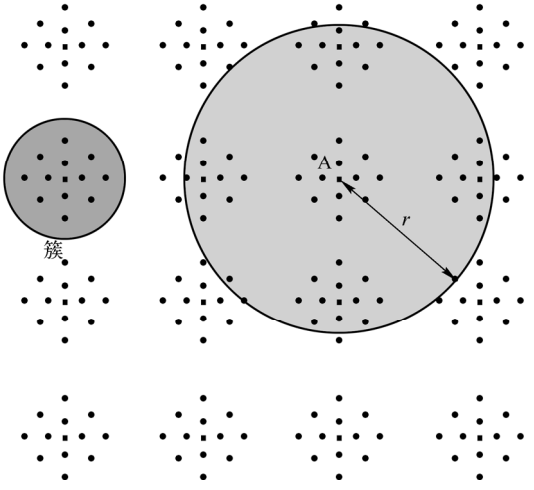


图 1.14　无线传感器/执行器分簇结构图[108]

Fig.1.14　Cluster of WSAN[108]

图 1.14 中的小圆代表簇的范围，也是传感器节点与执行器节点的通信范围，大圆代表执行器节点之间的通信范围。该算法详细分析并给出了单信道 AAC 的效用函数和时延函数。在阐述多信道 AAC 时，一个重要的区别是使用了两个独立的通信接口，如图 1.15 所示，每个执行器节点都有两个信道，C_s 用于相邻执行器节点通信，而 C_m 用于远距离通信。

执行器节点 A 直接通过 C_s 与相邻节点 B 通信，对于远距离的节点 D，需要通过多跳才能通信，因此节点 A 使用 C_m 利用节点 C 中转，虽然 C 和 D 是相邻节点，但考虑到信道切换的系统开销和时延，还是使用 C_m 进行通信。该算

法给出了多信道 AAC 的效用函数和时延函数。从实际仿真效果上看，多信道 AAC 的性能比单信道 AAC 好，特别是在负载变大的时候，由于单信道 AAC 策略中每个节点都参与转发数据，导致时延增加，而多信道 AAC 策略使用更大的传输范围，避免了累积时延。但是使用多信道 AAC 策略，每个节点必须知道相邻节点的准确位置，以便决定采用 C_s 还是 C_m 信道转发数据，这将增加节点信息维护成本，同时，一旦节点因移动发生位置变化，必须对全网节点广播位置更新消息，开销较大。

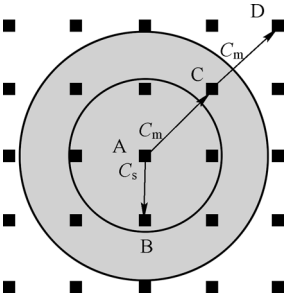


图 1.15 多信道 AAC 策略分析图[108]

Fig. 1.15 Multi-channel distributed AAC scheme analysis [108]

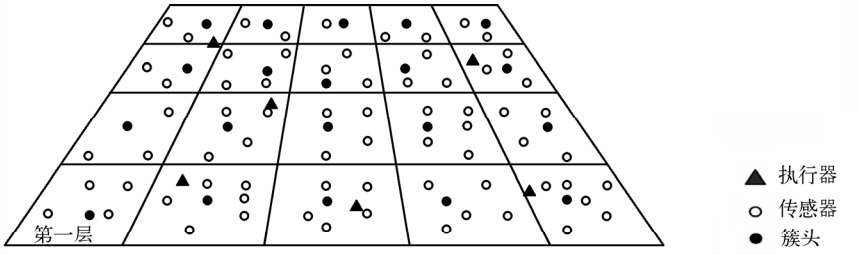
6．CM（Coordination Mechanism）算法[94]

CM 算法的主要贡献是提出了基于地理分簇范式的三层协作模型，在这个模型中，传感器节点是固定的，而执行器节点具有移动性能，如图 1.16 所示。

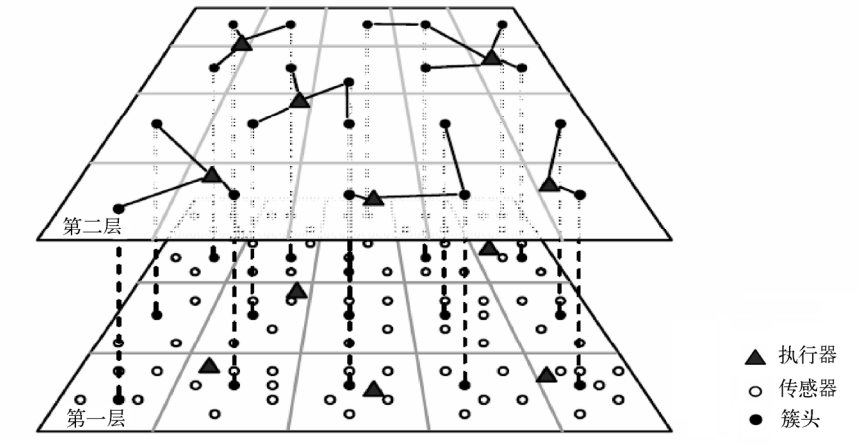
在 SS 协作层，主要考虑采用能量有效的手段采集数据，以最小化能耗和最大化生存期；在 SA 协作层，传感器节点以最小时延将数据传送到执行器节点；在 AA 协作层，执行器节点之间协作，有效而可靠地完成任务。该算法给出了执行器节点参与执行任务的条件：

$$\mathrm{ex}(N,A)=\alpha d(N,A)+\beta e(N)-\gamma n(A)+\delta p(A)$$

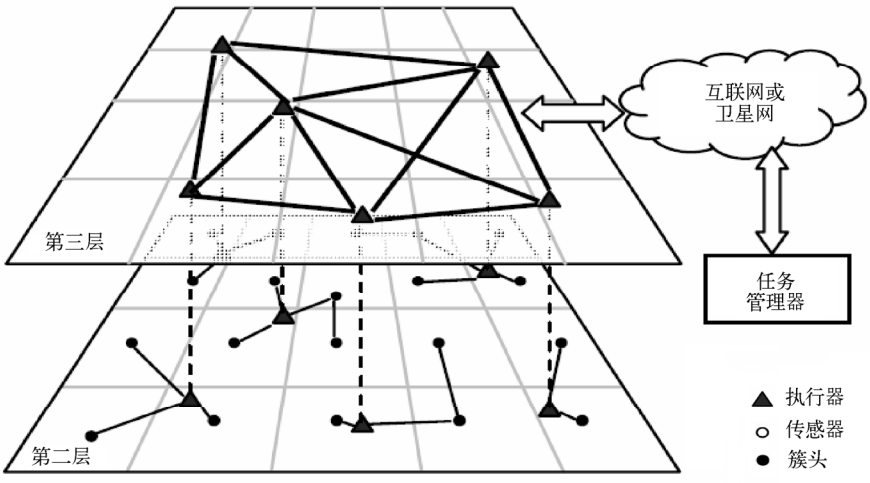
式中，$\mathrm{ex}(N,A)$ 表示执行器节点 N 参与执行 A 的期望；$d(N,A)$ 表示执行器节点 N 与执行区域的距离；$e(N)$ 表示节点 N 的剩余能量；$n(A)$ 表示执行任务 A 的节点个数；$p(A)$ 表示任务 A 的优先级，α、β、γ、δ 表示可调节参数。由于采用了分簇算法，网络被划分为以执行器节点为中心的分簇，避免了网络数据负载过大，同时解决了重叠区域执行器节点参与处理事件过多的问题。但是该模型给出的执行器节点参与执行任务的条件公式属于定性表达，并没有定量的计算公式，对于执行器节点执行任务的代价无法估计，其量纲也不统一，用于解决实际问题还有一定难度。



（a）传感器节点-传感器节点（Sensor-Sensor，SS）协作层



（b）传感器节点-执行器节点（Sensor-Actor，SA）协作层



（c）执行器节点-执行器节点（Actor-Actor，AA）协作层

图 1.16　三层协作模型[68]

Fig. 1.16　Three-level coordination model[68]

7．CPMA（Coordination Protocols of Multiple Actuator）算法[109]

Keiji Ozaki 等人提出的 CPMA 算法适用于多执行器多传感器网络模型（Multi-actuator/Multi-sensor，MAMS）。所谓的 MAMS 模型是指每个传感器节点都将数据传送给多个执行器节点，而每个执行器节点也接收来自多个传感器节点的数据。根据执行器节点的功能将协作机制分为四种：积极型（Active，AC）、半积极型（Semi-active，SA）、半消极型（Semi-passive，SP）和消极型（Passive，PS）。协议分为四个阶段：转发（forwarding）阶段、决策（decision）阶段、更新（update）阶段和执行（action）阶段。每个阶段采取的策略又分为两类：集中式和分布式。其中集中式是指网络中存在一个主执行器，所有决策都由主执行器发起，而其他执行器作为辅助；而分布式是指每一个执行器地位平等，独立执行任务。整个协作阶段可以用表 1.2 说明。

表 1.2 协作阶段表

Table 1.2 Coordination phase

	转发阶段	决策阶段	更新阶段	执行阶段
积极型（AC）	分布式	分布式	分布式	分布式
半积极型（SA）	分布式	分布式	集中式	集中式
半消极型（SP）	分布式	集中式	集中式	集中式
消极型（PS）	集中式	集中式	集中式	集中式

然而，在集中式机制中，整个网络由一个主执行器进行控制，一旦事件发生频繁，会造成数据等待，实时性能下降；而分布式机制中，需要多少执行器节点参与执行任务，并没有定量的标准，算法也没有给出集中式和分布式策略的各自开销，作者的另一文献[110]中也有类似阐述。

8．KE（Knowledge Exploitation）算法[111]

Martín López-Nores 等人设计了一种基于移动 ad hoc 网络（MANETs）的自治防火系统，与之前的大多数研究关注网络级议题如覆盖度、连通度、吞吐量，以及基于节点随机移动模型不同，系统提供了用于实际森林火灾的协作机制的质量评估平台，比较了受控机动性（controlled mobility）、预测技术（predictive techniques）、知识利用（knowledge exploitation）三类协作方法，其中比较前两种协作，知识利用技术扩大了执行器节点的移动自由度，节点可以在移动到指定区域外时也能轻易与其他节点建立联系，对网络的动态变化适应性最强，显示出良好的网络性能。但是知识利用协作方法关注的重点在于 MANETs 网络的节点移动性问题，对于任务的实时性、网络的能耗均衡等性能没有过多地考虑，

也没有相应的策略，并不适合作为无线传感器/执行器网络的协作框架。

9．DARA（Distributed Actor Recovery Algorithm）算法[112]

A.A.Abbasi 等人提出一种分布式执行器节点发现算法，主要针对执行器节点失效时，通过周围节点的协作，利用最少的执行器节点重新部署，构建网络的连通性。根据1连通和2连通的不同要求，算法又分为DARA-1C和DARA-2C。前一个算法设计的主要目的是选择一个合适的邻居节点代替死亡节点，整个重部署过程适用于那些由于某个邻居节点移动导致不连通的执行器节点。类似的DARA-2C 算法，用于修复 2 连通网络。每个节点都维护一张邻居节点位置表，利用发布心跳消息定期更新。当一个节点被检查到死亡，邻居节点启动修复程序，在所有邻居节点中选出节点度最小的顶替到死亡节点的位置，以最大程度减少对网络的影响；或者选出离死亡节点最近的节点，减少节点移动距离。当代替节点到达死亡节点位置后，通过消息广播自己的新位置，邻居节点重新计算网络的连通度，如果不满足 1 连通或者 2 连通，则重复算法直到到达连通度要求才停止。DARA 的主要优化目标是最小化节点重新移动的总距离和通信代价，以便使用最少的节点数量完成重新部署。整个发现过程是分布式的，无须外部条件网络自行修复。算法重点关注网络中节点死亡后如何修复连通度问题时，执行器节点之间的协作；对于网络工作正常的情况下，执行器节点如何对传感器节点的事件报告做出快速反应，并协调其他执行器节点采取行动并没有相关深入研究。

1.3.5 协作算法分类比较

无线传感器/执行器网络协作算法没有绝对的分类标准。从不同的技术角度考虑，具有多种不同的分类方式。而不同种类的协作算法具有各自不同的特点与应用场景，侧重点也不一样，没有绝对的优劣之分。

1．按照网络构架来划分（见表 1.3）

网络体系结构：传感器节点数据直接传送给执行器节点，还是通过执行器转发，分为自动或者半自动。

协作等级：协作分为传感器节点与传感器节点（Sensor-Sensor，SS）、传感器节点与执行器节点（Sensor-Actor，SA）、执行器节点与执行器节点（Actor-Actor，AA）。

节点的移动性：网络中的传感器节点和执行器节点是固定的还是移动的。

网络密度：指定区域内节点的数量和区域本身的大小的比值，分为不同等级。

表 1.3 按网络构架分类表

Table 1.3 Framework comparison

网络构架		典型协作算法								
		DCF	RCR	RCF	CA	AAC	CM	CPMA	KE	DARA
体系结构	自动	是	是	是	是	是	是	是	是	是
	半自动	否	是	否	否	否	否	是	否	否
协作等级	SS	否	是	是	否	否	是	否	否	否
	SA	是	是	是	是	是	是	是	是	是
	AA	是	否	是	是	是	是	是	是	是
移动性	传感器	固定	固定	固定	固定	固定	固定	固定	固定	移动
	执行器	固定	移动	移动	移动	移动	移动	固定	移动	移动
网络密度		密集	密集	密集	密集	密集	密集	密集	密集	密集

2．按照协作的支持技术来划分（见表 1.4）

协作以通信为目的，用来交换各节点的信息，需要以下技术支撑。

路由协议：通过协作通信，选出一条路由将数据传送到目的地。

定位技术：提供传感器节点和执行器节点的地理坐标位置，确定事件区域。

同步技术：有些网络采用分时数据传输，需要时间同步；有些任务执行，需要多个执行器协调一致采取行动，需要协作完成。

汇聚技术：减少多个传感器节点报告同一事件带来的网络负载。

分簇协议：将网络分为若干小的区域，分担网络载荷，避免过多的执行器节点参与同一事件的处理。

功率控制：不同的功率导致节点的无线信号覆盖范围发生变化。

表 1.4 按支撑技术分类表

Table1.4 Supporting technology comparison

支撑技术	典型协作算法								
	DCF	RCR	RCF	CA	AAC	CM	CPMA	KE	DARA
路由协议	地理	C-DEAR D-DEAR	地理	地理	地理	GAF	地理	地理	地理
同步技术	是	是	否	否	否	是	否	否	是
定位技术	是	是	是	是	是	是	是	是	是
汇聚技术	是	否	是	否	否	是	否	是	否
分簇协议	事件驱动	动态加权	事件驱动	否	是	异构地理	否	否	否
功率控制	是	否	是	是	是	否	否	是	否

3．按照具体应用要求来划分（见表 1.5）

协作算法必须考虑的具体应用要求如下。

实时性：有些应用对事件的响应时间有具体的要求。

事件频率：事件发生的频率和发生地点对执行器节点的响应时间是一个考验。

传输类型：根据传输模型分为连续型、事件驱动型和混合型三类。在连续型中，传感器节点按照一定的频率采集和发送数据；在事件驱动型中，传感器节点只有在事件发生后才传输数据；混合型就是上述两种类型的结合。

表 1.5　按应用要求分类表

Table1.5　Application requirements comparison

应用要求	典型协作算法								
	DCF	RCR	RCF	CA	AAC	CM	CPMA	KE	DARA
实时要求	低时延	有	低时延	低时延	有	有	有	低时延	有
事件频率	低	高	低	高	高	高	低	低	低
传输类型	事件驱动	事件驱动	事件驱动	事件驱动	事件驱动	事件驱动	事件驱动	事件驱动	混合

4．按照性能目标来划分（见表 1.6）

一些常用的算法性能指标成为协作算法设计必须考虑提高的参数。优化指标：在协作过程中，需要优化一个或者同时优化几个性能指标，如时延、能耗等等。规模：当节点增多，网络规模扩大时，算法是否具有适应能力。复杂度：计算的复杂度、所用的计算时间和需要占用的资源如 CPU、内存大小也是考察算法的依据。可靠性：算法本身的安全性和鲁棒性也是协作算法分类的标准，数据是否加密，是否具有容错能力，数据从传感器节点向执行器节点传输过程中，是否使用了知识学习和重传机制。

表 1.6　按性能目标分类表

Table1.6　Performance criteria comparison

性能目标	典型协作算法								
	DCF	RCR	RCF	CA	AAC	CM	CPMA	KE	DARA
优化指标	能耗时延	能耗时延	能耗时延	能耗时延	时延	能耗时延	容错性	覆盖率	连通度
网络规模	大	大	大	大	大	大	大	大	大
复杂度	无	无	无	无	无	无	无	无	DARA-1C O(N); DARA-2C O(KN)
可靠性	无	无	无	无	无	无	无	无	无

参 考 文 献

[1] Corson S. Macker J. Mobile Ad-hoc Networking (MANET): Routing Protocol Performance[S]. RFC 2501. 1999.

[2] Romer Kay Mattern F. The Design Space of Wireless Sensor Networks[J]. IEEE Wireless Communications. 2004,11 (6): 54-1.

[3] Haenselmann. Thomas. Sensor networks. GFDL Wireless Sensor Network textbook. [EB/OL] http://www.informatik.uni-mannheim.de/～haensel/sn_book.

[4] Hadim Salem Nader Mohamed. Middleware challenges and approaches for wireless sensor networks[J]. IEEE Distributed Systems Online. 2006,7 (3): 603-3001.

[5] Vassis D. Kormentzas G. Skianis C. Performance evaluation of single and multi-channel actuator to actuator communication for wireless sensor actuator networks[J]. Ad Hoc Networks. 2006,4(4):487-498.

[6] Petriu E. M. Georganas N. D. Petriu. Sensor-based information appliances[J]. IEEE Instrumentation and Measurement Magazine. 2000,3(4):31-35.

[7] Akyildiz Ian F. SuW. Sankarasubramaniam Y. Cayirci. A Survey on Sensor Networks.[J] IEEE Communications Magazine. 2002,40(8):102-114.

[8] Akyldiz F. Kasimoglu I. Wireless sensor and actuator networks: Research challenges[J]. Ad Hoc Networks Journal Elsevier. 2004,2(4):351-367.

[9] Melodia T. Pompili D. Gungor V. A distributed coordination framework for wireless sensor and actuator networks[C]. In Proceedings of the 6th ACM international symposium on mobile ad hoc networking and computing. 2005.341-352.

[10] Keini Ozaki Kenichi Watanabe Satoshi Itaya. A Fault-Tolerant Model of Wireless Sensor-actuator Systems[C]. In Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06). 2006:303-307.

[11] Arjan Durresi Vamsi Paruchuri. Geometric Broadcast Protocol for Heterogenous Sensor Networks[C]. In Proceedings of the 19th International Conference on Advanced Information Networking and Applications, AINA. 2005:1-343.

[12] S. Thrun M. Diel D. Ha. Scan alignment and 3-D surface modeling with a helicopter platform[C]. In: Proc. of the Int. Conf. on Field and Service Robotic. 2003.4:287-297.

[13] H. R. Everett D. W. Gage. A Third Generation Security Robot.[J] SPIE Mobile Robot and Automated Vehicle Control Systems. Boston, MA 1996. 29(03):118-126.

[14] Robotics DARPA Tactical Mobile. [EB/OL]http://www.darpa.mil/ato/ programs/tmr.htm2004.

[15] Sub-kilogram intelligent tele-robots (SKITs). [EB/OL]http://www.robotics.usc.edu/～behar/SKIT.html 2004.

[16] Ian F. Akyildiz Ismail H. Kasimoglu. Wireless sensor and actor networks: research challenges[J]. Ad Hoc Networks. 2004,2(4):351-367.

[17] Lesser V. Reflections on the nature of multi-agent coordination and its implications for an agent architecture[J]. Autonomous Agents and Multi-Agent Systems. 1998,1(1): 89-111.

[18] I. F. Akyildiz W. Su Y. Sankarasubramaniam. Wireless sensor networks: A survey[J]. Computer Networks. 2002, 38 (4):393-422.

[19] I. Chlamtac M. Conti J. N. Mobile ad-hoc networking:imperatives and challenges[J]. Ad Hoc Networks. 2003,1(1):13-64.

[20] A. J. Goldsmith S. Wicker. Design challenges for energycontrained ad-hoc wireless networks[J]. IEEE Wireless Communications. 2002,9 (4): 8-27.

[21] H. S. Kim T. F. Abdelzaher W. Minimumenergy asynchronous dissemination to mobile actors in wireless sensor networks[C]. In Proc. of the First ACM Int.Conf. on Embedded Networked Sensor Systems. 2003:193-204.

[22] M. Conti S. Giordano G. Maselli. Cross-layering in mobile ad-hoc network design[J]. IEEE Computer, Special Issue on AdHoc Networks. 2004, 37 (2): 48-51.

[23] B. Tavli W. Heinzelman. TRACE: Time reservation using adaptive control for energy efficiency[J]. IEEE Journal on Selected Areas of Communication. 2003,21 (10):1506-1515.

[24] M. Caccamo L. Y. Zhang L. An implicit prioritized access protocol for wireless sensor networks[C]. In: Proc. IEEE Real-Time Systems Symp. 2002:39-48.

[25] J. A. Stankovic T. F. Abdelzaher C. Real-time communication and coordination in embedded sensor networks[C]. Proceedings of the IEEE. 2003:1002-1022.

[26] Cardell-Oliver R. Smettem K. Kranz. Field testing a wireless sensor network for reactive environmental monitoring[C]. In Proceedings of the Intelligent Sensors, Sensor Networks and Information Processing Conference. 2004:14–17. 7-12.

[27] Szewczyk R. Osterweil E. Polastre. Habitat monitoring with sensor networks[J]. Communications of the ACM. 2004. 47 (6): 34-40.

[28] Santoni T. A. Santucci J. Using wireless sensor network for wildfi re detection.A discrete event approach of environmental monitoring tool[C]. Proceeding of ISEIMA 06. 2006:15-120.

[29] Szewczyk R Osterweil E. Polastre J. Habitat monitoring with sensor networks[J]. Wireless

sensor networks .2004,47(6):34-40.

[30] R. Polastre J. Design and implementation of wireless sensor networks for habitat monitoring[D]. 2003.

[31] Mainwaring A Polastre J. Szewczyk R. Wireless Sensor Networks for Habitat Monitoring[C]. Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. 2002:88-97.

[32] Cerpa A Elson J. Estrin D. Habitat monitoring: Application driver for wireless communications technology[J]. ACM SIGCOMM Computer Communication Review. 2001. 31:20-41.

[33] Fidelity and yield in a volcano monitoring sensor network.[EB/OL]. https://www.usenix.org/events/osdi06/tech/full_papers/werner-allen/werner-allen.pdf

[34] Hart J K Martinez K. Environmental Sensor Networks: A revolution in the earth system science[J]. Earth Science Reviews. 2006. 78(3-4): 177-191.

[35] Hu W Bulusu N. Chou C. Design and evaluation of a hybrid sensor network for cane toad monitoring[J]. ACM Transactions on Sensor Networks (TOSN) .2009,5(1):132-136.

[36] Milenkovi C A Otto C. Jovanov E. Wireless sensor networks for personal health monitoring: Issues and an implementation[J]. Computer Communications. 2006. 29(13-14):2521-2533.

[37] Bhargava A Zoltowski M. Sensors and wireless communication for medical care[C]. Proceedings of the 14th International Workshop on Database and Expert Systems Applications. 2003:956-962.

[38] Kim S Pakzad S. Culler D. Health monitoring of civil infrastructures using wireless sensor networks[C]. Proceedings of the 6th international conference on Information processing in sensor networks. 2007:254-263.

[39] Stankovic J A Cao Q. Doan T. Wireless sensor networks for in-home healthcare: potential and challenges[C]. High Confidence Medical Device Software and Systems Workshop. 2005:1312-1320.

[40] Otto C Milenkovic A. Sanders C. System architecture of a wireless body area sensor network for ubiquitous health monitoring[J]. Journal of Mobile Multimedia. 2006, 1(4);307-326.

[41] Shnayder V Chen B. Lorincz K. Sensor networks for medical care[C]. Proceedings of the 3rd international conference on Embedded networked sensor systems. 2005:314.

[42] 刘建. 无线传感器网络在军事的应用.[EB/OL]http://www.21eic.com/autocontrol/pdf/sensor091024m.pdf.

[43] SMART DUST Autonomous sensing and communication in a cubic millimeter [EB/OL]http:// robotics.eecs.berkeley.edu/～pister/SmartDust/.

[44] Cyrano 利用 IBM 的无线基础设施嗅出化学品的气味. [EB/OL]. https://www-900.ibm.com/cn/smb/solutions/wireless/othliv_cyrano_c.shtml.

[45] Sensor Webs. [EB/OL] http://sensorwebs.jpl.nasa.gov/.

[46] Tabar A M Keshavarz A. Aghajan H. Smart home care network using sensor fusion and distributed vision-based reasoning[C]. Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks. 2006:145-154.

[47] Chung W Y Oh S. J. Remote monitoring system with wireless sensors module for room environment[J]. Sensors & Actuators: B. Chemical. 2006. 113(1): 64-70.

[48] Ivanov B Zhelondz O. Borodulkin L. Distributed smart sensor system for indoor climate monitoring[C]. KONNEX Scientific Conference. 2002:10-11.

[49] Ferrari G Medagliani P. di Piazza. Wireless sensor networks: Performance analysis in indoor scenarios[J]. EURASIP Journal on Wireless Communications and Networking. 2007. 2007(1): 41.

[50] Pan M S Tsai C. H. Tseng. Emergency guiding and monitoring applications in indoor 3D environments by wireless sensor networks[J]. International Journal of Sensor Networks. 2006. 1(1): 2-10.

[51] S. Mahlknecht. Energy-self-sufficient wireless sensor networks for the home and building environment[M]. Institute of Computer Technology, Technical University of Vienna. Vienna, Austria. Dissertation thesis. 2004.

[52] Epstein. A. H. Milimeter-Scale,MEMS Gas Turbine Engines[C]. In Proceeedings of the ASME Turbo Expo 2003-Power for Land,Sea,and Air. 2003.

[53] 任丰原,黄海宁,林闯. 无线传感器网络[J]. 软件学报 Journal of Software. 2003. 14(07):1282-1291.

[54] Shih E Cho S. Ickes N. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks[C]. In: Proceedings of the ACM MobiCom 2001. Rome: ACM Press. 2001:272-286.

[55] Akyildiz I. F Su W. Sankarasubramaniam Y. Wireless sensor network: A survey[J]. Computer Networks. 2002,38(4):393-422.

[56] Werner-Allen G Swieskowski P. Welsh M. Motelab: A wireless sensor network testbed[C]. Information Processing in Sensor Networks IPSN 2005. Fourth International Symposium on. 2005:483-488.

[57] Ertin E Arora A. Ramnath R. A testbed for sensing at scale[C]. Information Processing in Sensor Networks, 2006. IPSN 2006. The Fifth International Conference on. 2006:399-406.

[58] Simulator-NS-2, The Network. [EB/OL]http://www.isi.edu/nsnam/ns/.

[59] 陈敏. OPNET 网络仿真[M]. 2004.

[60] Park S Savvides A. Srivastava M. SensorSim: a simulation framework for sensor networks[C]. Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems. 2000:104-111.

[61] Girod L Ramanathan N. Elson J. A software environment for developing and deploying heterogeneous sensor-actuator networks[J]. ACM Transactions on Sensor Networks. 2007,3(3):1-34.

[62] OMNET++Community Site. [EB/OL]In http://www.omnetpp.org/external/whatis.php.

[63] Zeng X Bagrodia R. Gerla M. a library for parallel simulation of large-scale wireless networks[J]. ACM SIGSIM Simulation Digest. 1998,28(1):154-161.

[64] Levis P Lee N. Welsh M. TOSSIM: Accurate and scalable simulation of entire TinyOS applications[C]. Proceedings of the 1st international conference on Embedded networked sensor systems. 2003:126-137.

[65] Karlof C, Wagner D. Secure routing in wireless sensor networks: Attacks and countermeasures[J]. Ad Hoc Networks. 2003, 1(2-3): 293-315.

[66] Undercoffer J, Avancha S, Joshi A, et al. Security for sensor networks[J]. Wireless networks. 2002, 8(5): 521-534.

[67] Perrig A, Stankovic J, Wagner D. Security in wireless sensor networks[J]. 2004.

[68] Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion: a scalable and robust communication paradigm for sensor networks[C]. Proceedings of the 6th annual international conference on Mobile computing and networking. 2000: 56-67.

[69] Sadagopan N, Krishnamachari B, Helmy A. The ACQUIRE mechanism for efficient querying in sensor networks[C]. Sensor Network Protocols and Applications, 2003. Proceedings of the First IEEE. 2003 IEEE International Workshop on. 2003: 149-155.

[70] Chu M, Haussecker H, Zhao F. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks[J]. International Journal of High Performance Computing Applications. 2002, 16(3): 293.

[71] Braginsky D, Estrin D. Rumor routing algorthim for sensor networks[C]. Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications. Atlanta, Georgia, USA: 2002: 22-31.

[72] Yu Y, Govindan R, Estrin D. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks[J]. UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023. 2001.

[73] Heinzelman W R, Chandrakasan A, Balakrishnan H. Energy-efficient communication protocol for wireless microsensor networks[C]. System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on. 2000: 10-12.

[74] Heinzelman W B, Chandrakasan A P, Balakrishnan H, et al. An application-specific protocol architecture for wireless microsensor networks[J]. IEEE transactions on wireless communications. 2002, 1(4): 660-670.

[75] Lindsey S, Raghavendra C S. PEGASIS: Power-efficient gathering in sensor information systems[C]. Aerospace Conference Proceedings, 2002. IEEE. 2002: 3-1125.

[76] Lindsey S, Raghavendra C, Sivalingam K. Data gathering in sensor networks using the energy*delay metric[C]. Parallel and Distributed Processing Symposium., Proceedings 15th International. 2001: 2001-2008.

[77] Manjeshwar A, Agrawal D P. TEEN: a routing protocol for enhanced efficiency in wireless sensor networks[C]. Parallel and Distributed Processing Symposium., Proceedings 15th International. 2001: 2009-2015.

[78] Manjeshwar A, Agrawal D P. APTEEN: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks[C]. Parallel and Distributed Processing Symposium., Proceedings International, IPDPS. 2002: 195-202.

[79] Luo H, Ye F, Cheng J, et al. TTDD: Two-Tier Data Dissemination in Large-Scale Wireless Sensor Networks[J]. Wireless Networks. 2005, 11(1): 161-175.

[80] Zehua Z, Xiaojing X, Xin W, et al. An energy-efficient data-dissemination protocol in wireless sensor networks[C]. World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006. International Symposium on a. 2006: 10-22.

[81] Wan C Y, Eisenman S B, Campbell A T. CODA: congestion detection and avoidance in sensor networks[C]. Proceedings of the 1st international conference on Embedded networked sensor systems. 2003: 266-279.

[82] Akan O B, Akyildiz I F. Event-to-actors reliable transport in wireless sensor networks[J]. 2005, 13(5): 1003-1016.

[83] Ee C T, Bajcsy R. Congestion control and fairness for many-to-one routing in sensor networks[C]. Proceedings of the 2nd international conference on Embedded networked sensor systems. Baltimore, Maryland, USA: 2004: 148-161.

[84] Rangwala S, Gummadi R, Govindan R, et al. Interference-aware fair rate control in wireless sensor networks[C]. Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications. 2006: 63-74.

[85] 鞠海玲, 崔莉, 黄长城. EasiCC:一种保证带宽公平性的传感器网络拥塞控制机制[J]. 计算机研究与发展. 2008(01): 16-25.

[86] 孙利民, 李波, 周新运. 无线传感器网络的拥塞控制技术[J]. 计算机研究与发展. 2008(01): 63-72.

[87] Pai-Hsiang H, Hwang A, Kung H T, et al. Load-balancing routing for wireless access networks[C]. INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE. 2001: 986-995.

[88] Hui D, Han R. A node-centric load balancing algorithm for wireless sensor networks[C]. Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE. 2003: 548-552.

[89] Ruihua Z, Lin W, Shichao G, et al. A Balanced Cluster Routing Protocol of Wireless Sensor Network[C]. Embedded Software and Systems Symposia, 2008. ICESS Symposia '08. International Conference on. 2008: 221-225.

[90] Liansheng T, Yanlin G, Gong C. A Balanced Parallel Clustering Protocol for Wireless Sensor Networks Using K-Means Techniques[C]. Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on. 2008: 300-305.

[91] Yanlin G, Gong C, Liansheng T. A Balanced Serial K-Means Based Clustering Protocol for Wireless Sensor Networks[C]. Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on. 2008: 1-6.

[92] Salehpour A A, Mirmobin B, Afzali-Kusha A, et al. An energy efficient routing protocol for cluster-based wireless sensor networks using ant colony optimization[C]. Innovations in Information Technology, 2008. IIT 2008. International Conference on. 2008: 455-459.

[93] Israr N, Awan I. Coverage Based Intercluster Communication for Load Balancing in Wireless Sensor Networks[C]. Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st International Conference on. 2007: 923-928.

[94] Jayashree L S, Arumugam S, Rajathi N. E/sup 2/LBC: an energy efficient load balanced clustering technique for heterogeneous wireless sensor networks[C]. Wireless and Optical Communications Networks, 2006 IFIP International Conference on. 2006: 7.

[95] Gupta G, Younis M. Performance evaluation of load-balanced clustering of wireless sensor networks[C]. Telecommunications, 2003. ICT 2003. 10th International Conference on. 2003: 1577-1583.

[96] He H, Yun X, Yu-E S, et al. Cluster-based load balancing multi-path routing protocol in wireless sensor networks[C]. Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on. 2008: 6692-6696.

[97] I-Shyan H, Cheng-Ching Y, Chiung-Ying W. Link stability, loading balance and power control based multi-path routing (SBPMR) algorithm in ad hoc wireless networks[C]. Telecommunications, 2003. ICT 2003. 10th International Conference on. 2003: 406-413.

[98] Ganjali Y, Keshavarzian A. Load balancing in ad hoc networks: single-path routing vs. multi-path routing[C]. INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. 2004: 1120-1125.

[99] Pham P, Perreau S. Multi-path routing protocol with load balancing policy in mobile ad hoc network[C]. Mobile and Wireless Communications Network, 2002. 4th International Workshop on. 2002: 48-52.

[100] Hu Y C, Johnson D B, Perrig A. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks[J]. Ad Hoc Networks. 2003, 1(1): 175-192.

[101] Hung L X, Hung L X, Sungyoung L. A coordination-based data dissemination protocol for wireless sensor networks[C]. Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004. 2004: 13-18.

[102] Qing H, Qing H, Yong B, et al. An Efficient Route Maintenance Scheme for Wireless Sensor Network with Mobile actors[C]. Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th. 2007: 155-159.

[103] Donghun L, Donghun L, Soochang P, et al. Continuous data dissemination protocol supporting mobile actors with a actor location manager[C]. Communications, 2007. APCC 2007. Asia-Pacific Conference on. 2007: 299-302.

[104] Haidong Yuan Huadong Ma Hongyu Liao. Coordination Mechanism in Wireless Sensor and Actor Networks[C]. Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06). 2006, 2:627-634.

[105] 胡四泉，李方敏，徐文君，刘新华. 无线传感器/执行器网络中能量有效的实时分簇路由协议[J]. 计算机研究与发展. 2008,45(1):26-33.

[106] 张丽. 无线传感执行网络中协作机制及算法的研究[D]. 硕士学位论文, 2007.

[107] Ortiz C. L Eric H. Structured Negotiation[C]. In ICMAS'02. 2002:1215 - 1222.

[108] Ortiz C. L Hsu E. Jardins M. Incremental Negotiation and Coalition Formation for Resource-bounded Agents[C]. In Proceedings of the AAAI Fall Symposium. 2001:116-122.

[109] Yadgar O Kraus S. Ortiz C. Hierarchical organizations for Real-time Large-scale Task and Team Environments[C]. In Proceedings of AAMAS'02. 2002: 1147 - 1148.

[110] Sandholm T Suri S. Improved Algorithms for Optimal Winner Determination in Combinatorial Auctions and Generalizations[C]. In National Conference on Artificial Intelligence(AAAI). 2002: 90-97.

[111] Soh L. K Tsatsoulis C. Satisficing Coalition Formation among Agents[C]. In Proceedings of the International Conference on Autonomous Agents and Multiagent System. 2001: 1062 - 1063.

[112] Soh L. k Tsatsoulis C. Real-time Satisficing Multiagent Coalition Formaiton[C]. In Working Notes of AAAI Workshop on Coalition Formaiton in Dynamic Multiagent Environments. 2002:7-15.

[113] Lesser V Horling B. Klassner F. BIG:An Agent for Resource-Bounded Information Gathering and Decision Making[J]. Artificial Intelligent Journal. 2000. 118:197-244.

[114] Horling B Lesser V. Vincent R. The Soft Real-time Agent Control Architecture[J]. Computer Science Technical Report. 2002:2-14.

[115] Tommaso Melodia Dario Pompili Vehbi C. A Distributed Coordination Framework for Wireless Sensor and Actor Networks. International Symposium on Mobile Ad Hoc Networking & Computing[C]. Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing. 2005:99-110.

[116] Melodia T. Pompili D. Gungor V. Communication and Coordination in Wireless Sensor and Actor Networks[J]. IEEE Transactions on Mobile Computing. 2007.6(10): 1116-1129.

[117] Melodia T. Pompili D. Akyldiz I. A Communication Architecture for Mobile Wireless Sensor and Actor Networks[C]. In Proceeding of IEEE International Conference on Sensor Mesh and Ad hoc Communications and Networks SECON. 2006:142-151.

[118] Dimitris Vassis George Kormentzas Charalabos Skianis. Performance evaluation of single and multichannel actor to actor communication for wireless sensor actor networks[J]. Elsevier B.V. 2005,4(4): 487-498.

[119] Keiji Ozaki Naohiro Hayashibara and Makoto. Coordination Protocols of Multiple Actuator Nodes in a Multi-Actuator/Multi-Sensor Model[C]. 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07). 2007: 62-67.

[120] Keiji Ozakia Tomoya Enokidob Makoto Takizawaa. Coordination protocols for a reliable sensor,actuator, and device network (SADN)[C]. In Proceeding of International

Conference on Complex, Intelligent and Software Intensive Systems. 2008: 193-199.

[121] M Lopez-Nores J. Garcia-Duque JJ Pazos. Qualitative assessment of approaches to coordinate activities of mobile hosts in ad hoc networks [J]. IEEE Communications Magazine. 2008. 12. 108-111.

[122] Ameer Ahmed Abbasi Mohamed Younis. Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks [J]. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. 2009. 20(9): 1366-1379.

[123] Shah G. A. Bozyigit M. Akan O. Real Time Coordination and Routing in Wireless Sensor and Actuator Networks[C]. In Proceedings of the 6th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking NEW2AN. 2006:365-383.

[124] Ngai C. H. Edith Lyu R. Michael Liu. A Real Time Communication Framework for Wireless Sensor-Actor Networks[C]. IEEE Aerospace Conference. 2006.2021-2029.

第 2 章　数据汇集树与动态交叉退避

按照网络的层次结构，可以将无线传感器网络路由协议分成两类：平面路由协议和分层路由协议。平面路由协议的代表是 Gossiping、Rumor routing、DD、COUGAR、ACQUIRE、SPIN 等，分层路由协议的代表是 LEACH、APTEEN、TEEN、TTDD、PEGASIS 等。在绝大多数平面路由协议中，如 DD、COUGAR、Gossiping、ACQUIRE、SPIN 等，以及部分分层路由协议中，如 TTDD、PEGASIS 等，都需要进行消息洪泛来建立路由。消息洪泛是指消息源节点产生消息，通过广播方式将消息发送给邻居节点，邻居节点收到消息后，继续用广播方式转发消息，最终使消息在全网内传播开。在洪泛消息中加入发送节点的地址信息，接收节点记录消息源地址，从而建立如图 2.1 所示的路由树。无线传感器网络数据汇集应用中，如果节点产生数据的速率一致（即传感器节点的数据采集周期一致，且不随时间变化），且所有节点在数据汇集过程中处于静止状态，可以通过构建负载均衡的数据汇集路由树的方式，均衡网络负载，延长网络寿命。本书将用于数据汇集的路由树称为数据汇集树。

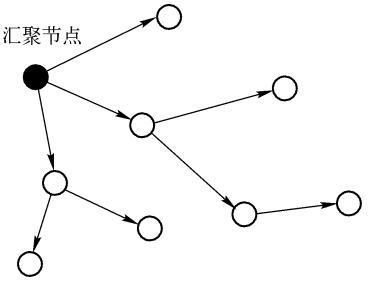


图 2.1　洪泛方式建立数据汇集树

Fig. 2.1　Establish of data gathering tree by flooding

一般情况下，无线传感器网络的通信协议被设计为独立的四层：物理层、数据链路层（或称 MAC 层）、网络层和应用层。路由算法用于为数据传输建立和维护路径，工作在网络层。退避机制用于避免消息碰撞，工作在 MAC 层。通常在数据汇集树的建立过程中，一般的路由算法都没有考虑 MAC 层的碰撞退避机制对数据汇集树的影响，但事实上，在洪泛过程中消息碰撞十分剧烈，并且 MAC 的退避策略和路由树的构造策略对构造数据汇集树的影响很大。因此，跨层设计在无线传感器网络通信协议的设计上非常必要[1~3]。

本章分析了碰撞机制对数据汇集树构造的影响，洪泛过程中的消息碰撞问题，提出了动态交叉退避窗口算法；在数据汇集树的路由瘫痪问题上，提出了定

义路由有效期和建立父节点优先级队列的策略。动态交叉退避窗口算法减少了在路由树构造过程中的消息碰撞，相比传统的退避算法，在本算法基础上构造的数据汇集树的拓扑结构更加合理（有更小的路径绕行值，并能够避开拥塞区域）。

2.1 数据汇集树的构造分析

2.1.1 退避机制对数据汇集树的影响

在无线通信系统中，退避机制通常被用于当多个设备同时发送消息时的碰撞避免，例如二进制指数退避（Binary Exponent Backoff，BEB）[4]。载波侦听多路访问碰撞避免（Carrier Sense Multiple Access with Collision Avoidance，CSMA/CA）算法作为无线通信典型的竞争随机访问信息机制，其接入信道的流程如图 2.2 所示。CSMA/CA 算法在发送一帧数据前，MAC 层首先随机退避一段时间 random(2^{BE}−1)，然后执行空闲信道评估（Clear Channel Assessment，CCA），如果此时信道为空闲则发送数据，否则采用二进制退避策略倍增窗口，继续退避直到信道空闲或者超时[5~7]。

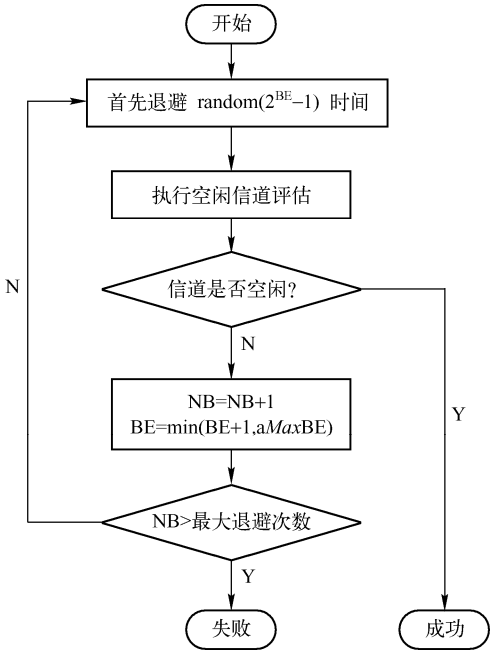


图 2.2 载波侦听多路访问碰撞避免算法

Fig. 2.2 CSMA-CA

在传统洪泛路由树的构造策略中（文献[8]），源节点将消息以广播包的方式发送出去，处于接收半径内的所有节点同时接收到消息。收到消息的节点通常会验证消息的重复性，如果不是重复消息就采用随机退避机制将消息通过广播方式转发出去，并同时记录消息源节点为其父节点。网络节点通常采用随机退避机制来避免消息碰撞，由于随机退避时，生成随机数的差异可能导致消息的传递并不是按照理想路径传播。因此，这种传统的路由树构造策略往往不能生成理想的数据汇集树。

如图 2.3（a）所示，消息源节点 A 产生一个带有本节点地址的消息，通过广播方式发送出去。节点 B 和节点 C 处于节点 A 的通信半径内，它们同时会收到节点 A 的消息，并设置其父节点为 A。然后节点 B 和节点 C 各自生成一个随机数用于发送数据时的碰撞避免。假设节点 B 所产生的随机数较大，而节点 C 产生的随机数较小（假如：BE_E=7, BE_C=1），因此节点 C 先把消息转发出去。处于节点 C 的通信半径内的节点 E 收到节点 C 的消息，设置其父节点为 C，同时产生了一个较小的随机数（假如：BE_E=2），导致节点 E 比节点 B 先将消息转发出去，节点 D 收到节点 E 的消息，并将节点 E 设置为父节点。这时构造的数据汇集树如图 2.3（a）所示。假设最优路由为最少跳数路由。显然，这种情况下生成的数据汇集树并不是最优的，因为消息从节点 A 传到节点 D 的跳数为 3，而最优（最短）跳数应该为 2，如图 2.3（b）所示。本书称图 2.3（a）的路径 A→C→E→D 为绕行路径。绕行路径会导致网络的负载分配不均衡，图 2.3（a）中节点 B 和节点 C 都处于汇集树的同一个层次上，但节点 C 需要转发节点 E 和节点 D 的数据，因此其负载大于节点 B 的负载。网络节点在退避失败的情况下（即在退避完成后，通过载波侦听方式发现信道仍然忙碌时），如果单纯增大初始碰撞退避窗口，就更加容易生成非优化的数据汇集树，使得各条路径的负载更加不均衡。因此，需要对传统的消息洪泛路由树构造策略进行改造，使之能够以较快的速度收敛成一棵负载均衡的数据汇集树。

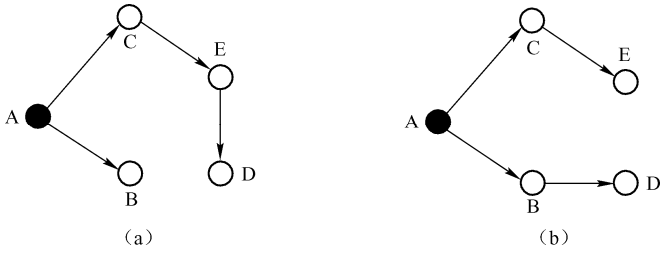


图 2.3　非优化数据汇集树（a）最优数据汇集树（b）

Fig. 2.3　Non-optimal data gathering tree (a) and optimal data gathering tree (b)

2.1.2 洪泛中的消息碰撞问题

由于在洪泛模型中，采用广播机制发送消息，处于发送节点周围众多的邻居节点同时收到消息，并同时转发消息。这样使得网络节点的活动趋于同步，同步的网络活动必然导致消息的碰撞概率增加。在无线传感器网络中，基于竞争的 MAC 层协议通常采用二进制指数随机退避机制实现信道共享，如 IEEE802.11MAC、802.15.4MAC。节点首先在退避窗口[0, CW]内等概率地随机选择一个退避数，用于发送消息时的碰撞退避。由于洪泛是以广播方式发送数据，无法启用 RTS/CTS、ACK 等信道预约和消息确认机制，因此碰撞概率就取决于 CW 值的大小。这种情况下，同时产生消息的节点数 n，在退避窗口 CW 情况下的碰撞概率计算公式为

$$\mathrm{Prob}(n,\mathrm{CW}) = 1 - \frac{\prod_{k=0}^{n-1}(\mathrm{CW}-k)}{\mathrm{CW}^n},\ 0 < n \leqslant \mathrm{CW} \qquad (2.1)$$

如图 2.4 所示，当 CW=100 时，10 个邻居节点的碰撞概率约为 35%，这种碰撞概率在无线传感器网络应用中通常是不能忍受的。

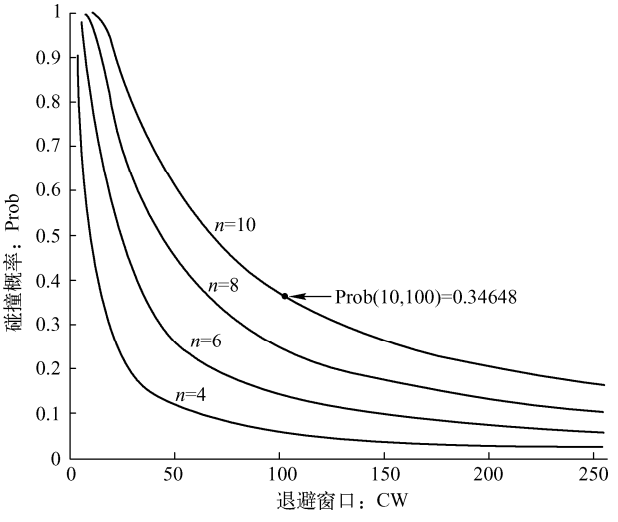


图 2.4 退避窗口与碰撞概率

Fig. 2.4 Contention window VS collision probability

在实际情形中，由于设备时钟不完全同步和代码执行速度的差异，碰撞情况与理论情况相比有差异。本书在基于 IEEE802.15.4 标准的 CC2430 EB & TI-MAC[9] 系统上进行了实验。实验过程中，其中一个节点发出一个请求消息，部署在其周

围（单跳范围）的其他节点收到请求消息后回复一个应答消息。反复此过程，记录丢包率。随着邻居节点数目的增加，丢包率增加，消息碰撞加剧，见表 2.1。

表 2.1　邻居数目与丢包率

Table 2.1　Number of neighbors versus packet loss rate

邻居节点数目	丢　包　率
2	7.5%
3	11.3%
4	20.2%
5	24.8%
6	30.5%
7	32.5%
8	35.6%
9	38.9%

2.1.3　数据汇集树的瘫痪问题

在传统的路由树构造过程中（如文献[10]），为使非优化的路由树得到修正，每个节点必须保存到数据源节点的最短跳数，并在所发送的每个消息中加入跳数信息（最短跳数加 1）。收到消息的节点判断消息中的跳数与当前跳数的大小。如果消息中的跳数比当前跳数更小，就修改当前跳数为消息中的跳数，并将跳数更新消息（加 1）后继续转发出去，同时修改父节点为当前接收消息源地址。如果消息中的跳数不小于当前跳数就忽略该消息。如图 2.5（a）所示，一开始由于随机退避策略的影响，导致生成了一棵非最优路由树。但最终节点 D 收到节点 B 的消息后，发现节点 B 为其更优（更少）跳数的父节点，则将父节点改变为节点 B，并更改自身跳数，然后将跳数更新信息转发出去。节点 F 和节点 G 收到 D 的新消息后，更新父节点信息和当前跳数，最终生成一棵优化路由树，如图 2.5（b）所示。

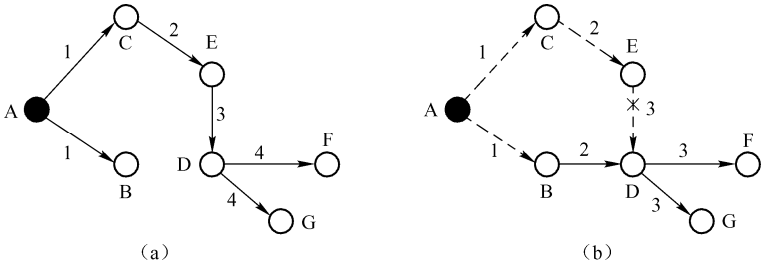


图 2.5　路由树的修正

Fig. 2.5　Revise routing tree

以上策略有两个缺点：

（1）由于跳数的更新，节点 D 及其后续节点都要重新转发洪泛消息，本书称这种现象为重复广播。重复广播会导致网络节点能量的浪费，更多的消息在网络中传输还会加剧网络的性能恶化，比如更多的消息碰撞和更大的时延。

（2）假设在建立好优化路由树后，由于某种原因导致节点 D 不能收到节点 B 的消息（比如节点 B 的电量耗尽或者出现障碍物等），但节点 D 保存的最小跳数为 2（其父节点为 B），所以始终不会“理睬”节点 E 发出的洪泛消息，这样就使得节点 D、F 和 G 的路由关系无法更新，而最终成为一棵瘫痪的路由树。

2.1.4 路径绕行评估与拥塞避免问题

相对较长的路径而言，数据在一条较短的路径上传输具有更小的延时和更少的能量消耗。因此，尽量缩短路径长度是建立数据汇集树的一个重要目标。假设最优数据汇集树定义为每个节点到执行器节点的距离均为最小跳数，有如下定义。

【定义 2.1】节点的数据传输路径绕行率（Circumambulate Rate，cr）定义为

$$\mathrm{cr} = \frac{h - H}{H} \tag{2.2}$$

式中，H 表示节点到执行器节点的最小跳数；h 表示节点到执行器节点的实际跳数。cr 值越小，表示节点到执行器节点的数据传输路径绕行程度越小，反之亦然。

【定义 2.2】具有 N 个节点的网络，其路径绕行率 CR 定义为

$$\mathrm{CR} = \sum_{i=1}^{N} \mathrm{cr}_i = \sum_{i=1}^{N} \frac{h_i - H_i}{H_i} \tag{2.3}$$

例如，在图 2.3（a）中，假设节点 A 为执行器节点，节点 D 到节点 A 的最小跳数应为 2 跳，而实际跳数为 3 跳，因此，节点 D 的数据传输路径绕行率为 $\mathrm{cr_D} = (3 - 2) / 2 = 0.5$。图 2.3（a）的网络路径绕行率为 $\mathrm{CR_a} = \mathrm{cr_B} + \mathrm{cr_C} + \mathrm{cr_D} + \mathrm{cr_E} = (1-1)/1 + (1-1)/1 + (2-2)/2 + (3-2)/2 = 0.5$。而图 2.3（b）的网络路径绕行率为 $\mathrm{CR_b} = (1-1)/1 + (1-1)/1 + (2-2)/2 + (2-2)/2 = 0$。由此可见，图 2.3（b）的数据汇集树比图 2.3（a）的数据汇集树优。

路径绕行率反映了数据汇集树的拓扑结构和负载的均衡情况。当 CR=0 时，表示每个节点到执行器节点的距离最短，此时形成一棵最短路径树（Shortest Path Tree，SPT）。理论上 CR 值越大，数据需要经过更多的跳数才能够传输给执行器节点，因此会产生更大的数据传输延迟和需要消耗更多的网络能量。而

在实际情况中，由于节点随机部署，可能造成某些区域节点密集分布，如图 2.6 所示的灰色区域。在这种情况下，密集区域内部相对于外部的消息碰撞更加剧烈，具有更大的丢包率，因此，处于该区域后方（相对于执行器节点）的数据流应该尽量绕开该区域，从而避免更多的碰撞和丢包。可见，这种情况下，一定程度的路径绕行是很有必要的。

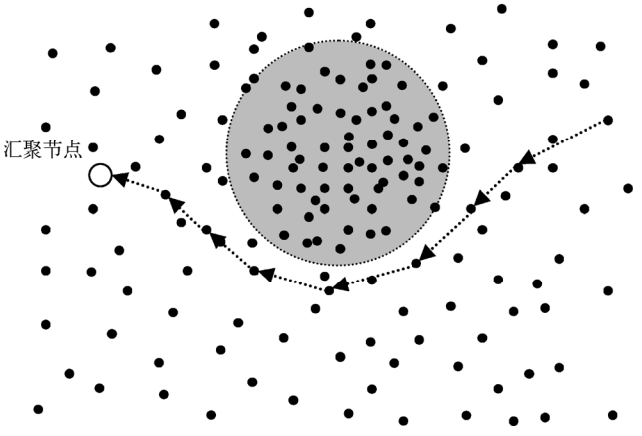


图 2.6　路由路径绕过密集区域

Fig. 2.6　Route path around density area

处于密集区域的节点相比其他区域的节点具有更多的邻居节点，因此，邻居节点的数目可以反映网络节点的密集程度，将 CR 的定义更改为如下：

【定义 2.3】节点的优化路径绕行率（Optimizing Circumambulate Rate，ocr）定义为

$$\mathrm{ocr}=\left(1+\frac{\max(n)}{M}\right)(1+\mathrm{cr})=\left(1+\frac{\max(n)}{M}\right)\left(\frac{1+h-H}{H}\right) \tag{2.4}$$

式中，n 表示邻居节点数；$\max(n)$表示到执行器节点的路径中最大的邻居节点数；M 为网络的最大节点邻居数，用于对 $\max(n)$作归一化处理。式（2.4）中，第一个“1”使得密度对 ocr 值成正比影响，第二个“1”使得密度对最短路径（此时 cr=0）也能产生影响。对于两条 cr 值相同的路径，如果其中一条穿过了网络节点较密集的区域，则其 ocr 值会增大，这样，ocr 值比 cr 值更能反映路径的拥塞情况。

【定义 2.4】网络的优化路径绕行率 OCR 定义为

$$\mathrm{OCR}=\sum_{i=1}^{N}\mathrm{ocr}_i=\sum_{i=1}^{N}\left(1+\frac{\max(n_i)}{M}\right)\left(\frac{1+h_i-H_i}{H_i}\right) \tag{2.5}$$

OCR 综合考虑了路径的绕行程度和节点的拥塞程度，体现了数据汇集树的负载均衡程度。OCR 值越小，数据汇集树的负载均衡程度越好，反之亦然。

2.2 动态交叉退避窗口算法

根据节点的通信半径，由执行器节点开始可以划分出每个节点所处的跳数等级。如图 2.7 所示，假设节点 A 为执行器节点，则其处于最低跳数等级（第 0 跳等级），此时节点 F 和节点 G 处于第 3 跳数等级。由此可知，造成图 2.3（a）路径绕行的原因是处于较高跳数等级的节点先于处于较低跳数等级的节点将消息广播出去。为了避免重复广播问题，解决的基本思想是让尽量处于前一跳数等级的节点洪泛完成之后，才让处于后一跳数等级的节点开始洪泛。为此，本书提出交叉退避窗口（Overlapping Backoff Window，OBW）。OBW 由两部分组成：固定退避时间（Fixed Backoff Time，FBT）和随机退避时间（Random Backoff Time，RBT）。OBW 计算公式为

$$\mathrm{OBW} = \mathrm{FBT} + \mathrm{RBT} \tag{2.6}$$

式中，RBT = Random() mod CW，Random()表示取随机数，CW 为竞争窗口（Contention Window）。CW、FBT、RBT 和 OBW 的单位为时隙或单位退避时间（Unit Backoff Slot, UBS）。图 2.8 为交叉退避窗口的工作示意图，RBT 的作用和传统的退避策略相同，即避免碰撞，而 FBT 的加入使得各个层次的退避时间错开，这样可以在一定程度上保证上一个跳数等级的节点洪泛完成之后，下一个跳数等级的节点才开始洪泛。在形式上，相邻跳数等级的退避窗口呈现交叉状，故称其为交叉退避窗口。相对于传统的退避窗口，交叉退避窗口虽然增加了时延，但其仅仅工作在汇集树的建立过程中，对网络的数据传输过程没有任何影响。

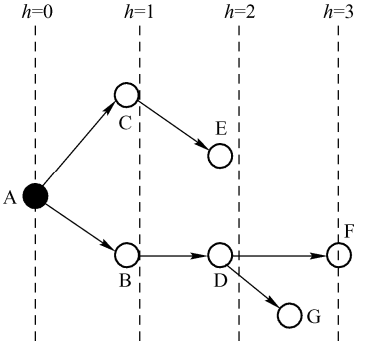


图 2.7　跳数等级

Fig. 2.7　Hop level

由图 2.8 可见，节点 B 和节点 C 为第 1 跳等级，其上跳等级仅有一个节点（执行器节点），节点 B 和节点 C 的 FBT 没有产生任何作用，因此，可以根据节点所处的跳数等级 h 计算 OBW，其计算公式为

$$\mathrm{OBW}(h)=\begin{cases}0, & (h=0)\\ \mathrm{RBT}, & (h=1)\\ \mathrm{FBT}+\mathrm{RBT}, & (h>1)\end{cases} \tag{2.7}$$

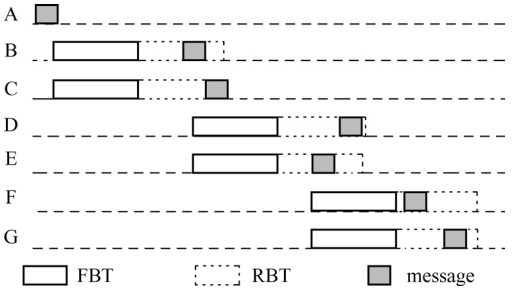


图 2.8　交叉退避窗口

Fig. 2.8　Overlapping Backoff window

如果节点的跳数等级为 0，表示该节点为执行器节点，无须任何退避时间。如果节点的跳数等级为 1，表示该节点为执行器邻居节点，无须固定退避时间。这样可以减少汇集树的构建时间。

根据式（2.7）和 RBT 的计算可知，交叉退避窗口的关键在于确定一个合理的 CW 值和 FBT 值。退避窗口值过大，则信道利用率减少；退避窗口值过小，则容易产生消息碰撞。因此，设置一个合理的退避窗口值非常重要。消息的碰撞概率与处于同一个碰撞区域的节点数和竞争窗口大小相关。一个处于高密度区域的节点，需要较大的 CW 值；而处于低密度区域的节点，其 CW 值可以相对较小。处于低密度区域的节点具有较小的 CW 值，这会使得其构建数据汇集树的速度大于高密度区域的节点，从而形成较长的路径。低密度区域路径较长，高密度区域路径较短，这符合拥塞避免原则。消息碰撞通常发生在直接相邻节点或者间接相邻节点（隐藏终端）同时发送消息的情况。在绝大多数无线传感器网络应用场景中，节点都是处于静止状态，因此每个节点的邻居节点数目基本保持不变。每个节点“偷听”周围节点一段时间的活动，或采用 Hello 机制进行邻居发现，就可以确定自身的邻居节点数目。邻居节点数目可以用于动态调整退避窗口值的大小，以适应不同密度区域的节点，从而提高了信道的利用率。本书称这种根据节点所处的区域密度动态设置的退避窗口为动态退避窗口（Dynamic Backoff Window，DBW）。假设给定一个可以承受的碰撞概率 α，根

据式（2.1）可以得到 DBW 计算公式如下：

$$\begin{aligned} &\mathrm{DBW}(n,h)=\min\{\mathrm{CW}\} \\ &\text{st.} \quad 1-\frac{\prod_{k=0}^{n-1}(\mathrm{CW}-k)}{\mathrm{CW}^{n}} \leqslant \alpha \end{aligned} \tag{2.8}$$

式中，n 表示节点的邻居节点数；h 表示节点的跳数。

因此， RBT = Random() mod DBW，DBW 计算的伪代码如下所示。图 2.9 显示了在碰撞概率 α=0.1 时，不同邻居节点的 DBW 值。

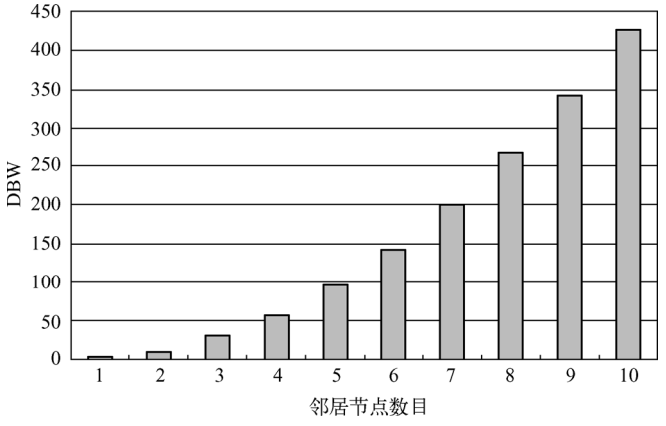


图 2.9　邻居节点数目与 DBW（α=0.1）

Fig. 2.9　Number of neighbors versus DBW(α=0.1)

动态退避窗口计算伪代码如下：

```
/*动态退避窗口计算*/
#define ALPHA 0.1 //定义网络可接受的碰撞概率
/*计算退避窗口函数*/
uint16_t GetDBWt(uint16_t n) {
   float32_t diff; //计算精度
   uint16_t dbw= n; //初始化退避窗口

   /*计算碰撞概率，并比较与期望值的偏差，从而得到退避窗口的大小。*/
   diff = getPro(dbw, n) - ALPHA;
   while( diff > 0.001) {
      dbw++;
      diff = getPro(dbw, n) - ALPHA;
   }

return (uint16_t)dbw;
}
```

由交叉退避窗口的原理可知，FBT 值的大小取决于父节点的 DBW 值。如果已知父节点已经退避的时间，就可以进一步缩短当前节点的 FBT 值，而父节点的 DBW 值可以通过洪泛消息附带告知子节点。因此，可以对 OBW 进行优化，提出动态交叉退避窗口（Dynamic Overlapping Backoff Window，DOBW），其计算公式为：

$$\mathrm{DOBW}_i(n,h)=\begin{cases}0, & h=0\\ \mathrm{RBT}_i, & h=1\\ \mathrm{DBW}_{\mathrm{parent}}-\mathrm{RBT}_{\mathrm{parent}}+\mathrm{RBT}_i, & h>1\end{cases} \tag{2.9}$$

式中，$\mathrm{DBW}_{\mathrm{parent}}$ 为节点 i 的父节点的退避窗口；$\mathrm{RBT}_{\mathrm{parent}}$ 为父节点已经退避的时间，$\mathrm{RBT}_i=\mathrm{Random}()\bmod \mathrm{DBW}_i$。一个节点在退避过程中可能收到几个父节点的 $\mathrm{DBW}_{\mathrm{parent}}$ 和 $\mathrm{RBT}_{\mathrm{parent}}$，此时选择最大窗口避免提前洪泛产生的路径绕行。

本书称在数据汇集树构建过程中使用动态交叉退避窗口的方法为动态交叉退避窗口（DOBW）算法。DOBW 算法的洪泛消息的包格式见表 2.2，其中包括标识节点的地址 SrcAddr、用于判断消息新旧程度的序号 Seq、跳数 Hop、节点动态退避窗口 DBW 和已退避时间 RBT。初始情况下，由于节点不知道邻居节点的情况，需要设置一个默认的邻居节点数目 N_{Init}，N_{Init} 可以根据不同的应用环境进行配置。

表 2.2　DOBW 算法洪泛消息包头格式

Table 2.2　Packet header of DOBW

数　据　域	SrcAddr	Seq	Hop	DBW	RBT
长　　度	16bit	8bit	8bit	16bit	16bit

DOBW 算法的伪代码如下所示。

```
/*DOBW 算法*/
MsgHandle(pmsg){
  if (msg->dstAddr == LOCAL_ADDR){ //本地消息
    rendermsg(pmsg); //向应用层提交数据
    forwardMsg(pmsg); //转发消息
  }
  else{ //洪泛消息
    NeighborNumber = updateNeighbor(pmsg->srcAddr，NeighborNumber); //获取邻居节点数目
    tmpNumber= NeighborNumber;
  if (is_new_msg(pmsg->seqno)){
    remember_msg(pmsg); //保存消息关键字
    Hop = pmsg->hop + 1;
```

```
            FBT = pmsg->DBWpartent – pmsg->RBTparent;
            DBW = getBackoffSlot (NeighborNumber, Hop));//计算动态退避窗口大小
            RBT = Random() mod DBW;
            Backoff();//MAC 层进行退避，同时“偷听”信道
        }

        if(msg_from_another_father(pmsg)){
            FBT = FBT - pmsg->RBTparent;
            newRBT = Random() mod getBackoffSlot(tmpNumber --, Hop) ;
            RBT = min(RBT, newRBT);
            Backoff();//MAC 层进行退避，同时“偷听”信道
        }
    else return;
    }
    }

    BackoffDoneHandle(pmsg){
      updateMsg(pmsg); //更新消息中的关键域
      forwardMsg(pmsg);//转发消息
    }
```

由于洪泛属于网络层功能，而碰撞退避属于 MAC 层功能，因此 DOBW 算法属于跨层设计方法。当需要使用洪泛功能时，就将 MAC 层切换到 DOBW 算法，否则执行传统退避。

2.3 路由瘫痪防止策略

在防止路由瘫痪方面，本书提出通过定义路由有效期和建立优先级父节点队列两个策略来解决。

2.3.1 定义路由有效期

为实现对瘫痪路由的修复，每个节点不但保存当前到执行器节点的最短跳数，并且在发送的每个消息中加上数据汇集树的有效期（Duration），一旦超过 Duration 时间，就认为该数据汇集树失效，此时设置该节点到执行器节点的跳数为无穷大。执行器节点以 Duration 为周期不断地刷新数据汇集树。该策略一方面有效地避免了数据汇集树瘫痪后继续传输数据造成网络能量的浪费，使得网络可以适应一定程度上的拓扑变化，为支持节点移动通信提供了一定的前提条件；另一方面，Duration 信息的加入，可以作为传感器节点执行任务的设定

时间，从而更加灵活地对传感器任务和功耗进行控制。Duration 值的设置，需根据具体的应用需求，综合考虑能耗和网络拓扑变化速度等因素。例如，在流域水质监测应用场景中，由于拓扑比较固定，可以设置 Duration=60min。在社区安防应用场景中，如果使用了移动节点，可以设置 Duration=30s。

2.3.2　建立优先级父节点队列

当网络出现瘫痪时，只有执行器节点的下一次消息洪泛才能重新建立有效的数据汇集树。在网络瘫痪期间，任何沿着瘫痪分支的反馈数据都不能传回到执行器节点，如图 2.10 所示。

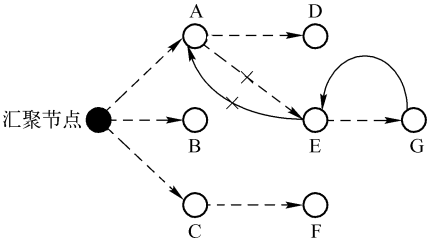


图 2.10　瘫痪的数据汇集树

Fig. 2.10　Paralysis of data gathering tree

因此，本书提出为每个节点建立一个基于优先级的父节点队列。为了避免路径回环，父节点集合描述如下：

$$F(u)=\{x\,|\,\mathrm{hop}(x)<\mathrm{hop}(u)\} \tag{2.10}$$

式中，$F(u)$为节点 u 的父节点集合；x 为节点 u 的邻居节点；hop 为节点的跳数。

在每个洪泛消息中，增加节点的当前剩余能量和任务量，并为每个节点建立一个基于优先级的父节点队列，其优先级按照剩余能量和任务量综合考虑。在数据沿着汇集树反向传播时，每次数据传输加上 ACK 确认机制。ACK 机制的加入，一方面增加了数据传输的可靠性；另一方面，如果出现路由瘫痪，数据包超过最大重发次数（典型值为 3 次），则认为当前父节点出现故障，节点自动降低当前父节点的优先级为最低，选择次优先级的父节点进行数据传输，并产生路由更新信息，通过广播消息转发出去。这样就保证了在路由出现瘫痪情况时，数据能够正常传输，并修复瘫痪路由树。如图 2.10 所示，当节点 A 和节点 E 之间的链路出现故障时，节点 E 收不到节点 A 的 ACK 消息，节点 E 将其父节点 A 的优先级降到最低，同时选取次优先级节点 B 为其当前父节点。数据通过节点 B 得以可靠传输，从而修复了瘫痪的路由，如图 2.11 所示。

父节点的优先级计算方法如下：

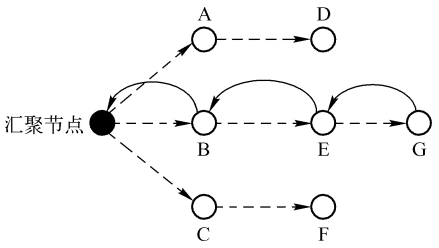


图 2.11　瘫痪路由的修复

Fig. 2.11　Revise the break routing

$$\mathrm{Pri}(e,T)=\alpha e+\beta T,\alpha+\beta=1 \text{ 且 } \alpha,\ \beta\in[0,1] \qquad (2.11)$$

式中，e 为当前剩余能量；T 为当前节点任务量；α、β 为权重。

通常情况下，剩余电量可以表示节点处理任务的多少，如节点剩余电量少时，可以认为处理过的任务较多。但是，如果供电方式不同就不能简单地把任务量与剩余电量联系在一起，并且任务量的大小与网络性能密切相关（如丢包率、网络时延以及负载均衡程度等），故在优先级计算时加入任务量。任务量可以用处于路由缓冲队列中的待处理消息数来估计。实际应用中，e、T 需要经过归一化处理。根据具体应用环境中对数据传输速度和能量的要求，相应地调整 e、T 的权重 α、β。如果出现优先级相等的情况，按照接收消息的先后顺序排队。例如，一个农业种植大棚环境监控的应用场景中，对丢包率和实时性要求不高，而对能量有效性要求较高，可以设置 α=0.6，β=0.4。

2.4　仿真验证

仿真实验平台为 TOSSIM。设置网络节点数为 100，其中利用执行器节点充当汇聚节点（Sink）。节点部署情况如图 2.12 所示，最左边为执行器节点，中间区域的节点分布较密集，形成一个拥塞区域（congestion area）。在拥塞区域中部，设置了一个测试节点（test node）。Radio model 采用 Fixed radius(10.0)，设置距离因素 Distance scaling factor 为 0.7，网络可接收碰撞概率 α=0.05，单个退避时间槽 UBS=32 个符号（symbols）时间。

仿真过程中，使用 Hello 数据包（hello count）进行邻居发现，图 2.13 显示了测试节点的邻居（neighbors）发现情况，经过 3 次洪泛后，测试节点准确地获得了 11 个邻居节点。图 2.14 显示了测试节点的 DBW 值变化情况，DBW 值的变化趋势与当时获得的邻居节点数目一致。图 2.15 显示了测试节点的退避（backoff counter）计数值变化情况，可见，退避窗口在动态地改变。

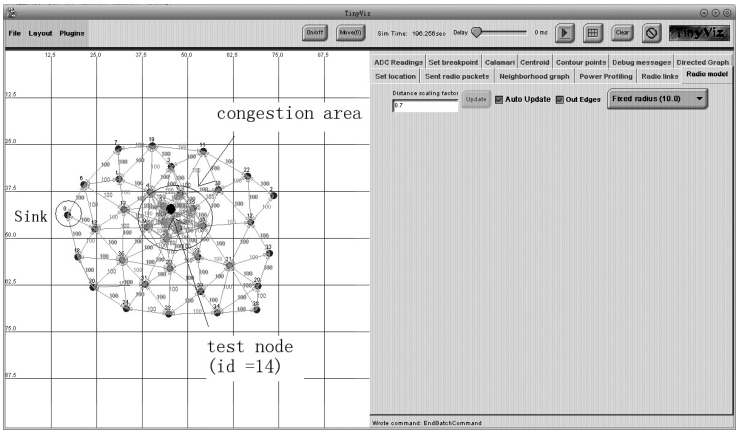


图 2.12　节点部署情况

Fig. 2.12　Deployments of nodes and connection status

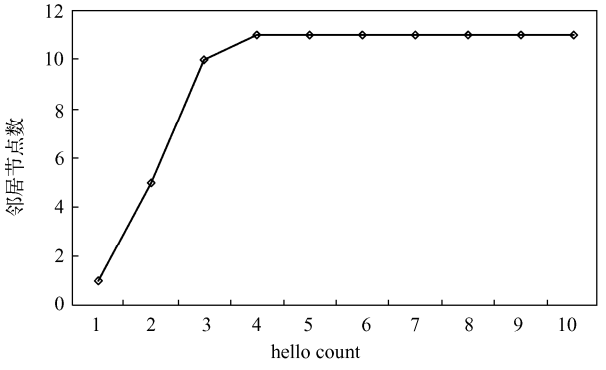


图 2.13　测试节点邻居个数

Fig. 2.13　Number of neighbors detected by test node

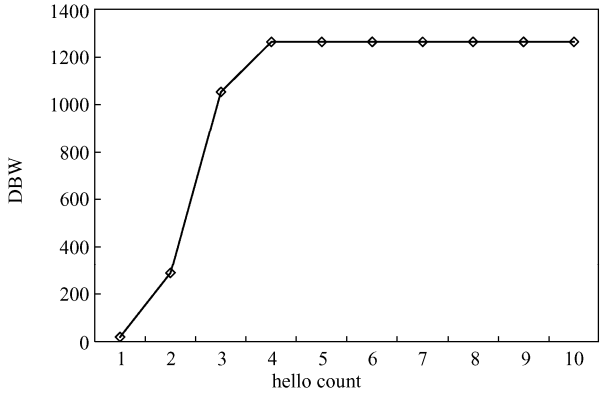


图 2.14　测试节点 DBW 的变化情况

Fig. 2.14　DBW of test node

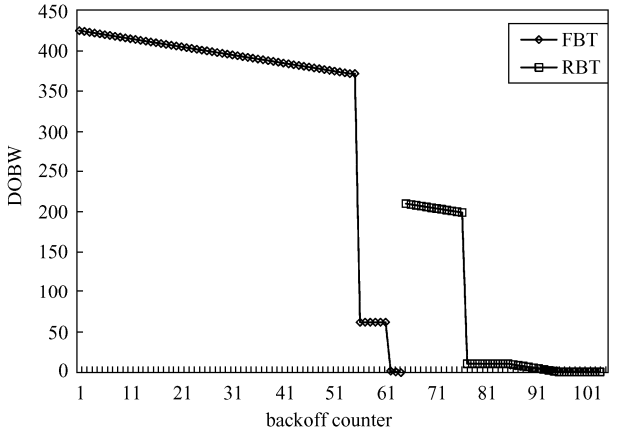


图 2.15　测试节点的 DOBW 变化情况

Fig. 2.15　Process of DOBW of test node

图 2.16、图 2.17 和图 2.18 分别显示了 802.11、802.15.4 和 DOBW 生成数据汇集树的情况。从 DOBW 生成的数据汇集树的形态上可以直观地看出，DOBW 算法在绕行程度和拥塞区域避免上都较 802.11 和 802.15.4 优。

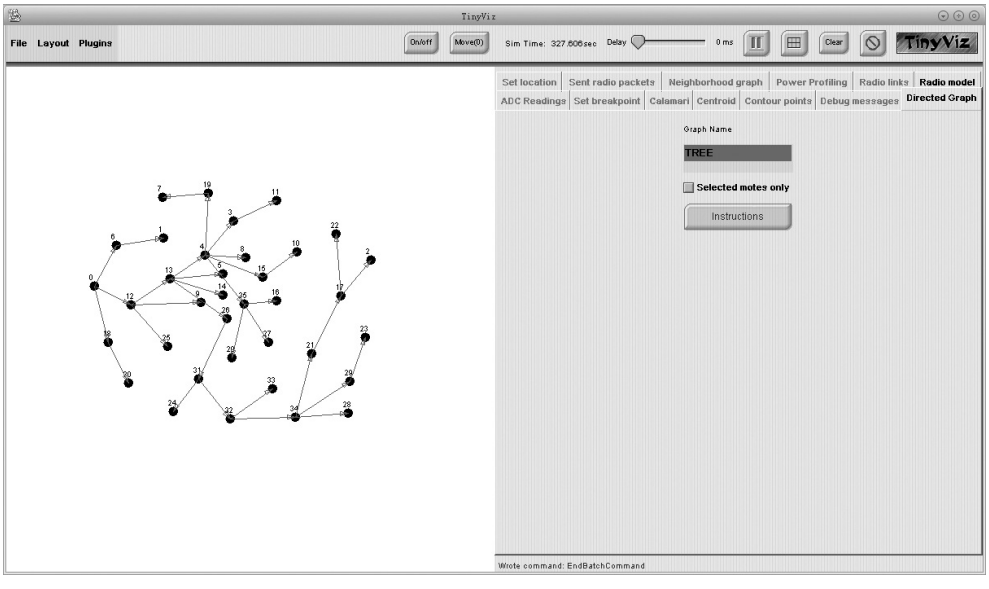


图 2.16　802.11 生成的汇集树

Fig. 2.16　Routing tree based on 802.11

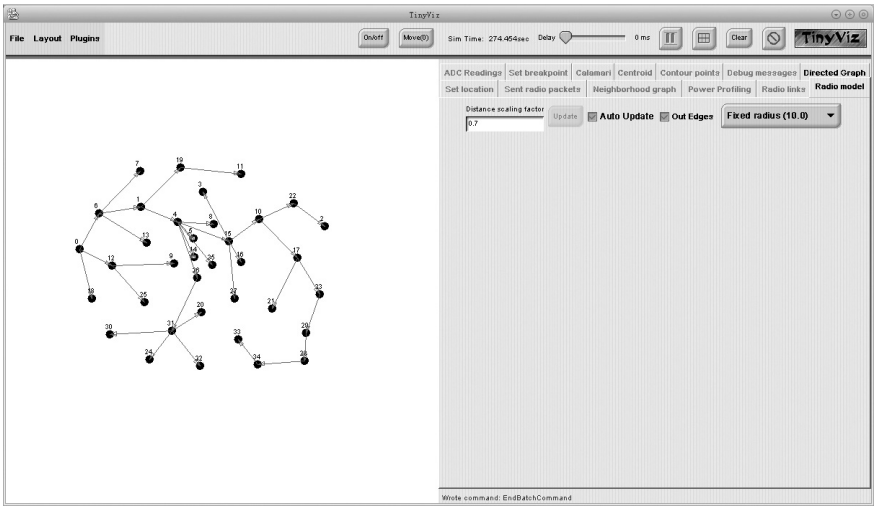


图 2.17　802.15.4 生成的汇集树

Fig. 2.17　Routing tree based on 802.15.4

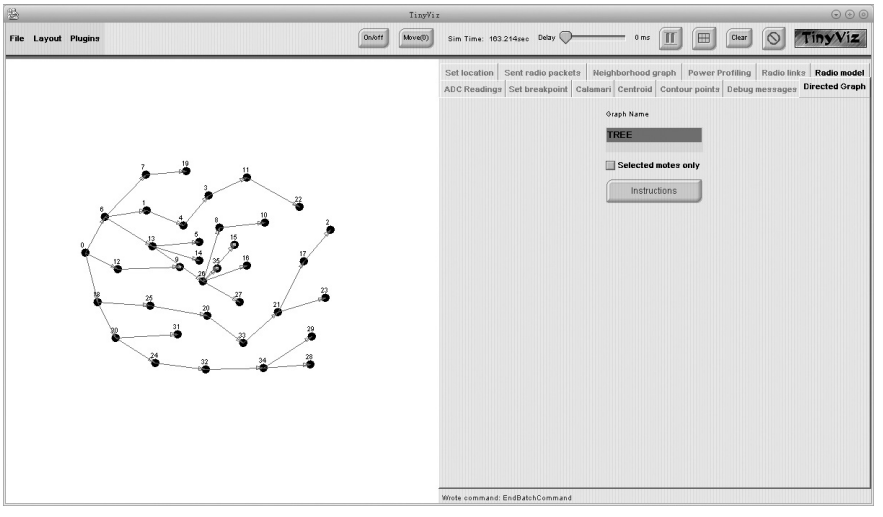


图 2.18　DOBW 生成的汇集树

Fig. 2.18　Routing tree based on DOBW

由于节点自身不能检测碰撞的产生，因此，本书利用 TOSSIM 仿真环境的全局控制方式，建立全局碰撞表。每当节点发送数据时，查询全局碰撞表，判断邻居节点和间接邻居节点的状态。如果邻居节点或间接邻居节点处于发送状态，则认为发生了一次碰撞。

图 2.19 显示了 802.11、802.15.4 和 DOBW 三种算法在洪泛（flooding）过程中的消息碰撞情况。可见，802.11 和 802.15.4 的碰撞（collisions）情况相当，

而 DOBW 的碰撞次数最小，其原因在于动态退避窗口的大小是根据节点邻居数目确定的，因此能够保证较小的碰撞概率。经计算，本书提出的 DOBW 相比 802.11 和 802.15.4 碰撞减少约 76.3%。图 2.20 显示了这三种算法的 OCR 值，OCR 值反映路径优化程度，OCR 值越小，路径优化程度越好，从图 2.20 可以看出本书提出的 DOBW 算法在路径上具有明显的优化，经计算，相比 802.11 和 802.15.4 路径优化程度提高了约 76.1%。

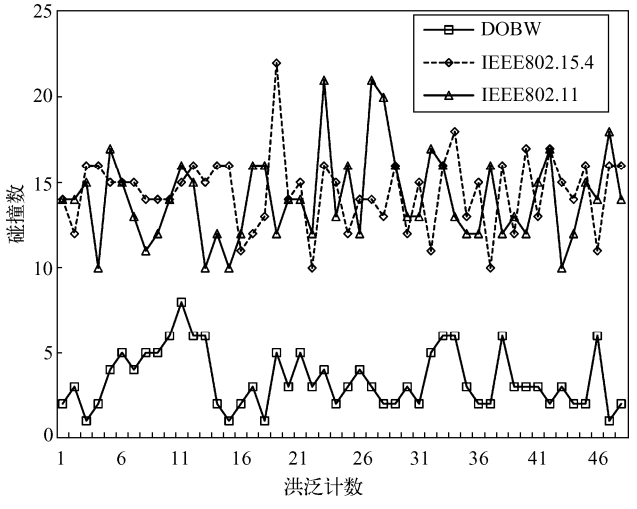


图 2.19　网络的碰撞测试

Fig.2.19　The collisions of network

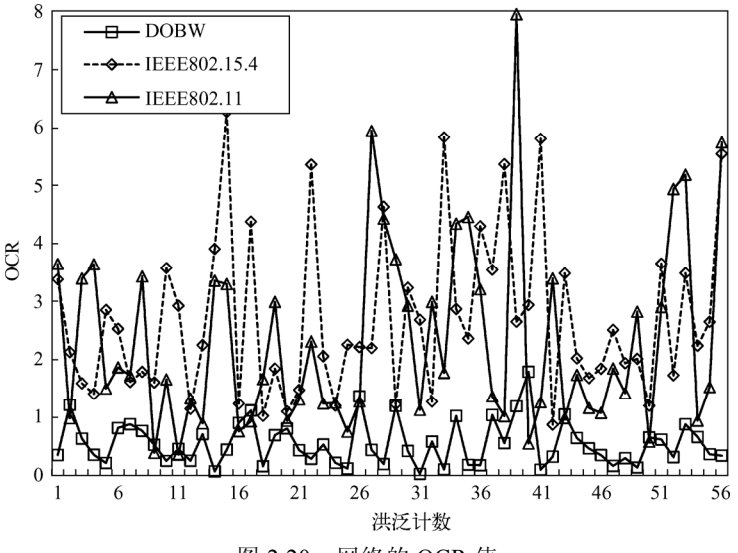


图 2.20　网络的 OCR 值

Fig. 2.20　OCR of network

2.5　小结

在无线传感器网络树状拓扑结构中，通常采用洪泛方式建立数据汇集树。本章分析了在洪泛过程中退避机制对构建数据汇集树的影响，洪泛过程中的消息碰撞问题、数据汇集树的瘫痪问题，以及路径绕行和拥塞避免问题，提出了动态交叉退避窗口算法（DOBW）。仿真实验表明，相比 802.11 和 802.15.4，DOBW 算法显著地减少了洪泛时的消息碰撞，显著提高了数据汇集树的路径优化程度，为构建一棵负载均衡的数据汇集树打下了基础。

参 考 文 献

[1] Seung-Joon C, Ki-Hyuk K, Sang-Jo Y. An Efficient Cross-Layer Based Flooding Algorithm with Retransmission Node Selection for Wireless Sensor Networks[C]. Advanced Information Networking and Applications - Workshops, 2008. AINAW 2008. 22nd International Conference on. 2008: 941-948.

[2] Yu-Pin H, Kai-Ten F. Cross-layer routing for congestion control in wireless sensor networks[C]. Radio and Wireless Symposium, 2008 IEEE. 2008: 783-786.

[3] Wang C, Li B, Sohraby K, et al. Upstream congestion control in wireless sensor networks through cross-layer optimization[J]. 2007, 25(4): 786-795.

[4] Committee I C S L. IEEE Std 802.11 Wireless LAN Medium Access Control (MAC) and Physical (PHY) specifications[S]. 1999.

[5] Cali F, Conti M, Gregori E, et al. IEEE 802.11 protocol: design and performance evaluation of anadaptive backoff mechanism[J]. IEEE journal on selected areas in communications. 2000, 18(9): 1774-1786.

[6] Zheng J, Lee M J. A comprehensive performance study of IEEE 802.15. 4[J]. Sensor network operations. 2006: 218-237.

[7] Lu G, Krishnamachari B, Raghavendra C S. Performance evaluation of the IEEE 802.15. 4 MAC for low-rate low-power wireless networks[J]. Proc. EWCN. 2004, 4.

[8] 彭刚, 曹元大, 钟伟军, et al. 无线传感器网络的数据汇聚机制[J]. 计算机工程. 2006(06): 115-117.

[9] TI CC2430 data sheet[EB/OL].http://focus.ti.com.cn/cn/docs/prod/folders/print/cc2430.html.

[10] 刘湘雯, 侯惠峰, 胡捍英. 无线传感器网络基于树的能量有效路由协议[J]. 计算机应用研究. 2007(01): 294-296.

第 3 章　静态负载均衡数据汇集树生成算法

根据无线传感器网络不同的应用环境，可以将网络数据流分为三种类型：连续型、查询驱动型和事件驱动型[1]。在连续型网络数据流中，各个传感器节点源源不断地将数据向执行器节点汇集，本书称这种应用为数据汇集应用。在数据汇集应用中，由于数据流量大、多点向一点汇集，相邻路径之间容易发生串扰、信道竞争和冲突，造成多次重传，严重时就会引发拥塞。拥塞不但会增加丢包率，影响传输可靠性，还会浪费宝贵的能量资源。预防和缓解网络拥塞是负载均衡需要解决的首要问题。负载均衡是预防和缓解拥塞的重要手段。负载均衡对网络性能影响很大，不但可以提高带宽的利用率和节点能量的利用率，还能降低网络的传输延迟，减少网络丢包，提高数据传输的成功率。

文献[2]阐述了无线传感器网络可能发生两种类型的拥塞：一种是节点级拥塞，节点产生数据报文的速度超过了自身发送报文的速度，导致缓存队列变长，增大了排队延时，甚至产生队列溢出，造成重传；另一种是链路级拥塞，无线信道是共享信道，当多个相邻节点同时竞争使用无线信道时，就会产生访问冲突，由冲突产生的重传会增加分组的服务时间，降低链路利用率和网络的吞吐量，引起链路级拥塞。本书认为，在无线传感器网络中还存在第三种拥塞，即热点拥塞。由于较多的节点将数据发送到一个转发节点（如网关节点）时，该转发节点就成为网络的一个热点，如果此热点接收到的分组速率过快，就会造成转发队列增加，甚至溢出，从而产生热点拥塞。本书在假设不发生节点级拥塞的情况下，尝试通过调整拓扑结构来均衡流量分配，从而减少访问冲突，避免链路级拥塞和热点拥塞。

文献[3]指出，在树状路由中，节点仅仅需要维护自身与其父节点的链路状态，无须维护庞大的路由表。相比网状（Mesh）路由，树状路由的存储开销和计算开销都较小。因此，树状路由最适合低成本、低功耗、资源非常受限的无线传感器网络数据汇集应用。无线传感器网络数据汇集应用中，如果节点产生数据的速率一致（即传感器节点的数据采集周期一致，且不随时间变化），且节点部署后位置不再变化时，可以通过构建负载均衡的数据汇集树的方式，均衡

网络负载。在网络节点资源和通信能力相同的情况下，最短路径树（Shortest Path Tree，SPT）的拓扑结构通常具有转发跳数少和传输延迟低的优点。而一个连通图生成的最短路径树通常不止一棵，在无线传感器网络数据汇集应用中，不同的最短路径树具有不同的负载均衡情况。因此，本章旨在尝试研究无线传感器网络数据汇集应用中的最短路径数据汇集树的负载均衡问题，提出一种无线传感器网络最短路径负载均衡数据汇集树建立算法（Load-balanced Data Gathering Tree based on SPT，LDGT-SPT）。LDGT-SPT 算法主要分为三大步骤：第一步，通过邻居发现构造相对邻居图，并通过层次发现构造层次图；第二步，采用度小优先原则生成一棵最短路径树；第三步，通过流量均衡策略，进一步规避局部热节点，均衡网络数据流量，生成负载均衡树。3.3 节通过理论分析，证明了 LDGT-SPT 算法的正确性，即算法收敛，并能构造一棵最短路径负载均衡树。3.4 节的仿真实验表明，相比相关算法，LDGT-SPT 算法使得网络性能有显著提高。

3.1　LDGT-SPT 算法思想

在无线传感器网络数据汇集应用中，数据经过多跳方式，多点向一点（执行器节点）汇集。相对于执行器节点，处于上游的节点需要承担转发下游节点数据的工作，这使得上游节点的负载大于下游节点的负载，形成所谓的“漏斗效应”。“漏斗效应”是无线传感器网络数据汇集应用固有的特点，且不同的“漏斗”形状存在不同的负载均衡程度，如图 3.1 所示。从直观上看，图 3.1（a）的数据流集中程度小于图 3.1（b）和图 3.1（c），因此可以认为图 3.1（a）的负载均衡程度较好。通常情况下，网络负载均衡程度较好时，网络的丢包较小、吞吐量较大，也不易发生拥塞，网络寿命会得到相应的延长。在不能改变节点通信能力的情况下，“漏斗效应”不可避免，但可以通过构造合理的“漏斗”形状均衡网络节点的负载。LDGT-SPT 算法的目标就是构造一个最大负载均衡程度的“漏斗”形状来收集网络数据。

对于随机部署在检测区域的无线传感器节点，假设每个节点的初始电量和通信能力（通信半径）相同。在网络开始运行阶段，每个节点发送 Hello 消息（该消息包含本地节点地址）用于邻居发现。经过数次 Hello 消息之后，每个节点就可以准确获得与自身相邻且能够通信的节点，称为邻居节点。然后由执行器节点开始，采用动态交叉退避算法进行层次发现，每个节点就可以知道自身距离执行器节点的跳数或层次。通过邻居发现和层次发现，所有节点的链路连

接起来就可以形成一个无向层次图。该无向层次图是由分布在网络各处的节点及链路形成的。由于没有任何地方集中保存该图的拓扑信息，因此只能依靠分布式算法进行处理。LDGT-SPT 算法的基本思想是采用分布式算法，由一个无向层次图生成一棵最短路径负载均衡的数据汇集树。LDGT-SPT 算法过程如图 3.2 所示，负载均衡程度的定义将在后面给出。

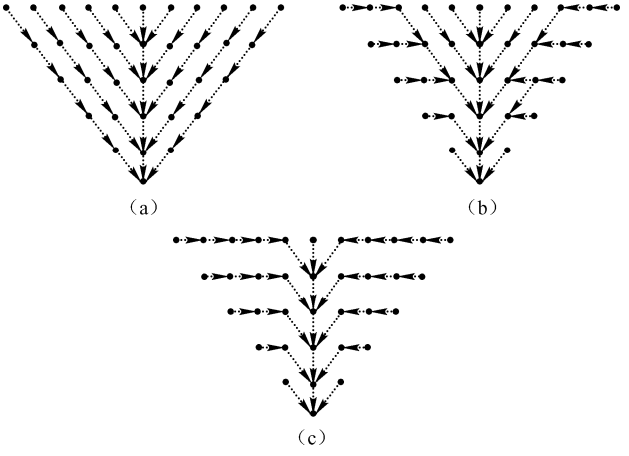


图 3.1　漏斗效应

Fig. 3.1　Tunnel effect

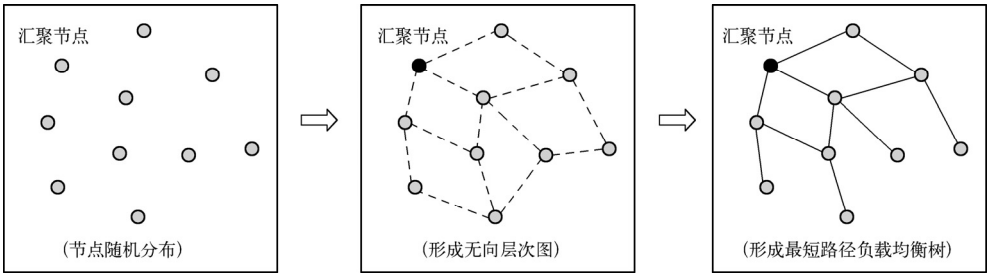


图 3.2　LDGT-SPT 算法过程

Fig. 3.2　Process of LDGT-SPT

LDGT-SPT 算法的关键在于如何从一个无向层次图构建一棵最短路径负载均衡数据汇集树。该算法采取以下两个步骤：首先采用节点的度作为参考因素，构建一个初始最短路径数据汇集树；然后对此最短路径数据进行修正，进一步规避局部热节点，均衡网络数据流量，最终生成一棵负载均衡数据汇集树（负载均衡程度最大）。修正过程中，节点统计自身作为树根的子树的大小（子节点个数），了解各个父节点子树的大小，并通过改变自身所属的父节点，均衡父节

点的子树大小，从而使网络负载均衡分配。负载均衡树是在最短路径树的基础上修正得到，因此，生成的负载均衡树为最短路径负载均衡数据汇集树。

3.2　LDGT-SPT 算法描述

3.2.1　相关定义

无线传感器网络最短路径负载均衡数据汇集树的建立问题可以归结为图论中的生成树问题。

【定义 3.1】对无线传感器网络的图模型有如下定义：

（1）无线传感器网络所有节点随机分布在二维平面上，节点位置固定，通信半径相同，形成一个简单连通无向图 $G=(V, E)$，其中 V 表示无线传感器网络节点集，E 表示节点对称链路形成的边集。

（2）每个节点具有唯一标号，并且可以排序。节点 x 的标号记为 $\mathrm{id}(x)$，即对于 $\forall x$，$\forall y, x \in \mathrm{V}$，$y \in \mathrm{V}$，满足 $\mathrm{id}(x) < \mathrm{id}(y)$或者 $\mathrm{id}(x) > \mathrm{id}(y)$。

（3）节点与其邻居节点连接的边数称为节点的度数。节点 x 的度数记为 $d(\mathrm{x})$，$\mathrm{d}_x(y)$表示节点 x 的邻居 y 的度数。

（4）节点 x 到执行器节点的最小跳数记为 $h(x)$。$h_x(y)$表示节点 x 的邻居 y 到执行器节点的最小条数。

（5）节点 x 的邻居节点集记为 $N(x)$。

（6）节点 x 的父节点集记为 $F(x)$，$F(x)=\{u \mid u \in \mathrm{V} \cap u \in N(x) \cap h(u)= h(x)-1\}$。

（7）以节点 x 为根的子树中的节点个数记为 $t(x)$。$t_x(y)$表示以节点 x 的邻居 y 为根的子树中的节点个数。

（8）传感器节点产生数据的周期为 T。

$N(x)$和 $d(x)$可以通过邻居发现得到，$h(x)$可以通过动态交叉退避算法进行层次发现得到，$t(x)$初始化为 1，在数据汇集的过程中，节点通过统计上行数据流的源节点地址个数得到 $t(x)$，并通过 ACK“稍带”的方式告知邻居节点。为了获得邻居节点的 t 值，节点必须“偷听”网络数据包。

对于一棵数据汇集树，假设其深度为 N，给定层次 i，$i<N$，设从 $i+1$ 到 N 层的所有节点总共有 M 个，则这 M 个节点产生的数据流必然全部通过第 i 层。假设第 i 层上有 n 个节点，其 t 值分别为 $t_1, t_2, \ldots, t_n$，且 $t_1+t_2+\ldots+t_n=M$。为了均衡网络数据流，$t_1, t_2, \ldots, t_n$ 之间的差值应该最小化。根据乘法原理，当 $t_1 \times t_2 \times \ldots \times t_n$ 最大时，$t_1, t_2, \ldots, t_n$ 之间的差值最小，因此有如下定义：

【定义 3.2】由图 $G=(V, E)$生成的一棵最短路径树 $T=(V, E')$，$E' \subseteq E$，深度为 N，第 i 层的节点个数为 n_i，则树 T 的第 i 层的负载均衡程度定义为

$$W_L(i)=\frac{\prod_{j=1}^{n_i} t_{ij}}{\bar{t}_i^{n_i}} \tag{3.1}$$

式中，t_{ij} 为第 i 层第 j 个节点的 t 值；$\bar{t}_i$ 为第 i 层节点的平均 t 值，$\bar{t}_i$ 反映理论最大程度负载均衡的情况。因此，$W_L(i)$ 越大，表示该层节点的负载均衡程度越大。

【定义 3.3】树 T 的负载均衡程度定义为

$$W_T=\sum_{i=1}^{N} W_L(i) \tag{3.2}$$

式中，W_T 越大，表示数据汇集树的负载均衡程度越大。

例如，图 3.4、图 3.5 和图 3.6 是由图 3.3 所示的无向层次图生成的 3 棵树（生成树 T_1、生成树 T_2 和生成树 T_3）。根据定义 3.3 可知，这 3 棵树的负载均衡程度分别为 6.625-e3、3.0-e2 和 6.25-e2，可见，生成树的负载均衡程度为 $T_3<T_2<T_1$。从直观上看，其拥塞情况与之相符。

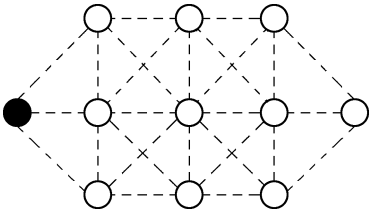


图 3.3　无向层次图

Fig. 3.3　Undirected level graph

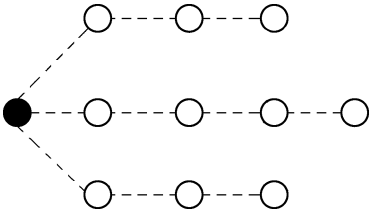


图 3.4　生成树 T_1（W_T≈3.76035）

Fig. 3.4　Spanning tree T_1（W_T≈3.76035）

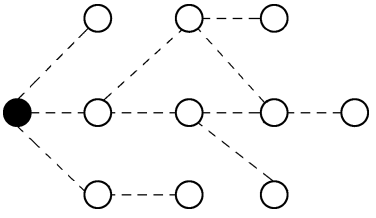


图 3.5　生成树 T_2（W_T≈2.85175）

Fig. 3.5　Spanning tree T_2（W_T≈2.85175）

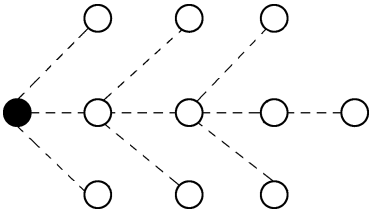


图 3.6　生成树 T_3（W_T≈2.45375）

Fig. 3.6　Spanning tree T_3（W_T≈2.45375）

【定义 3.4】由图 $G=(V, E)$生成的一棵最短路径树 $T=(V, E')$，$E' \subseteq E$，如果树 T 的负载均衡程度最大，则称 T 为最短路径负载均衡数据汇集树。

在无线传感器网络数据汇集应用中，传感器节点周期性地产生数据，注入网

络，数据经多跳方式流向执行器节点，数据流的大小和方向相对稳定，其拥塞往往发生在热节点处。因此，从直观上理解，可以认为度数越大的节点热度越高。

【定义 3.5】对于图 $G=(V, E)$的任意节点 $x\in V$，$y\in V$，定义运算符$\prec$对 x，y 排序，其排序规则为

$$x\prec y\begin{cases}d(x)<d(y),\ d(x)\neq d(y)\\ id(x)<id(y),\ d(x)=d(y)\end{cases},\ x\in V,\ y\in V \tag{3.3}$$

【定理 3.1】通过运算符$\prec$，对于 $G=(V, E)$中的任意节点 u，其父节点可以构造一个全序集（$F(u)$，$\prec$）。

【证明】根据定义 3.1，由于每个节点具有唯一标号，并且可以排序，所以定理 3.1 显然成立，证明从略。证毕。

为了构造最短路径负载均衡数据汇集树，LDGT-SPT 算法定义以下四种消息：

Request_Msg(x, y)表示节点 x 请求加入 y 作为其子节点，消息包含节点 x 的 t 值。

Confirm_Msg(y, x)表示节点 y 确认了节点 x 的请求消息或更改消息，并“稍带”自身更改后的 t 值。

Change_Msg(x, y, z)表示节点 x 请求加入 y 作为其子节点，并脱离父节点 z，消息包含节点 x 的 t 值。

Notify_Msg(x, y)表示节点 x 通知父节点 y，节点 x 的 t 值变化量。当发送队列有数据要发送时，可以取消该消息，t 值可以通过“稍带”的方式发送。

节点随时“偷听”邻居节点的网络消息，实时更新邻居节点和自身的 t 值。

3.2.2　LDGT-SPT 算法流程

LDGT-SPT 算法流程如下：

Step0：开始，节点初始化，$d(x)=0$，$h(x)=0$，$N(x)=\Phi$，$F(x)=\Phi$，$t(x)=1$；

Step1：节点发送 Hello 消息进行邻居发现，构造相对邻居图，使每个节点获得 $N(x)$、$d(x)$和所有的 $d_x(y)$。

Step2：由执行器节点开始，采用动态交叉退避算法进行层次发现，使每个节点获得 $h(x)$，以及所有的 $h_x(y)$。

Step3：若 $h(x)=1$，则节点 x 为执行器节点的邻居节点，则直接建立与执行器节点的父子关系。否则，节点使用运算符$\prec$，构造一个全序集（$F(x)$, $\prec$），并选择全序集（$F(x)$, $\prec$）的第一个节点 y 为父节点。同时，发送 Request_Msg(x, y)消息通告，收到 Request_Msg 的节点 y，发送 Confirm_Msg(y, x)消息确认。节点 x 收到确认消息后建立与 y 的父子关系，最终生成一棵最短路径数据汇集树。

Step4：在数据汇集的过程中，统计上行数据流的源节点地址个数，得到每个节点的 t 值，并通过 ACK“稍带”的方式，告知邻居节点。经过一个数据采集周期 T，可以获得确定的 $t(x)$和所有的 $t_x(y)$。

Step5：采用流量均衡算法，进一步规避局部热节点，具体操作如下：

（1）对于节点 x，其父节点为 u，如果$\exists v$，满足$\{v \in F(x) \bigcap t_x(u) > t(x)+t_x(v)\}$，则发送 Change_Msg($x$, u, v)消息，并将父节点设置为 v，否则执行 Step6。

（2）收到 Change_Msg(x, u, v) 消息的节点 u 和 v，分别改变自身的 t 值，并使用 Notify_Msg 消息或者数据“稍带”方式通知上层节点。

（3）收到 Notify_Msg 消息或者数据“稍带”的 t 值变化的节点，继续通知上层节点，直到与执行器连接的顶层节点。

（4）执行 Step5①。

Step6：结束。

LDGT-SPT 算法主流程的伪代码如下所示。

```
/*LDGT-SPT 算法主流程部分*/
Main() {
  d=0，h=0，NeighborList=NULL，t=1，state=HELLO_PENDING    //初始化
  switch(state) {
  case HELLO_PENDING:
    SendHello();
    /*在消息接收中断中更新 NeighborList */
    HelloTimerStart();
    /*在定时器中断中更改 state=LAYER_FINDING */
    break;
  case LAYER_FINDING:
    DOBW(); //执行动态交叉退避窗口算法
    SortFartherList(); //对 NeighborList 中的父节点进行排序，并选出初始父节点
    state=DATA_GATHERING;
  case DATA_GATHERING:
    TimerStart(T, PERIOD_MODEL); //启动周期性定时器，数据采集周期为 T
    /*在定时器中断中采集并传输数据*/
    break;
  }
}
```

定时器（Timer）主要用于控制每个过程的时间和数据采集周期，其伪代码如下所示。

```
TimerHandle() {
  switch(state) {
```

```
        case HELLO_PENDING:
            SendHello();
            if (HelloCount < HELLO_TIMES)
               HelloTimerStart();
            else {
            state=LAYER_FINDING;
            if(LOCAL_ADDR==执行器)
               SendLayerMsg();                    //由执行器节点开始发起层次发现
            }
            break;
         case DATA_GATHERING:
            TimerStart(T, PERIOD_MODEL);          //启动周期性定时器，数据采集周期 T
            SensorStart();                        //采集数据
            SendDate();                           //发送数据
            LoadBalance();                        //根据算法 Step5①进行流量均衡
            break;
         }
      }
```

消息接收部分伪代码如下所示。

```
      LDGT-SPT 消息接收部分
      MsgHandle(msg){
         switch(msg->type) {
         case HELLO:
            UpdateNeighbor (msg);                 //更新邻居节点信息
            break;
         case Layer_Msg:
            UpdateParents (msg);/                 //更新父节点信息
            forward(msg);                         //转发数据给下一个节点或应用端口
            break;
         case DATA:
            forward(msg);                         //转发数据
            break;
      case Request_Msg:
            UpdateChildrens (msg);                //更新子节点信息
            if (IsMychild) {
               UpdateT(msg);                      //更新 t 值
               SendConfirm();
               SendNotify();                      //通知上层节点
            }
            break;
         case Confirm_Msg:
            UpdateParents (msg);                  //更新父节点信息
```

```
            break;
          case Notify_Msg:
            UpdateChildrens (msg);              //更新子节点信息
            if (IsMychild) {
                UpdateT(msg);                   //更新 t 值
                SendNotify();                   //继续通知上层节点
            }
            break;
        }
    }
```

3.3 LDGT-SPT 算法举例与理论证明

3.3.1 LDGT-SPT 算法举例

节点随机部署在监测区域中，通过 LDGT-SPT 算法 Step1 和 Step2 建立如图 3.7 所示的相对邻居图，并发现各自距执行器节点的层次，虚线表示邻居关系。节点的 id(*h*, *d*)如图 3.7 所示。

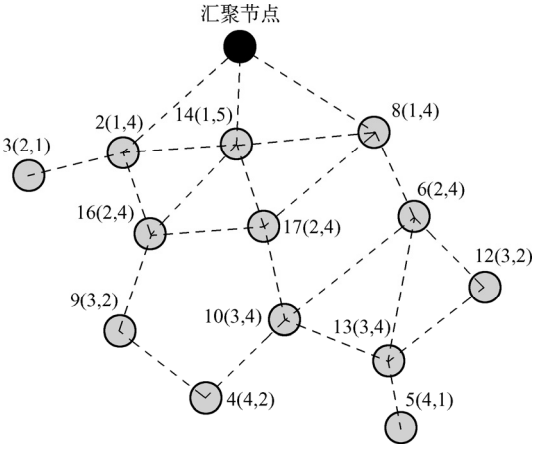


图 3.7　无向层次图

Fig. 3.7　Undirected level graph

由算法的 Step3，子节点选择度最小的父节点作为当前父节点，生成如图 3.8 所示的最短路径数据汇集树，实线表示父子关系、节点的 id(*h*, *d*, *t*) 如图 3.8 所示。例如，16 号节点使用运算符$\prec$，构造一个全序集(F(16), $\prec$)={2, 14}，则选择 2 号节点为其父节点，并建立父子关系。经计算，此时构造的数据汇集树的负载均衡程度约为 2.6042。

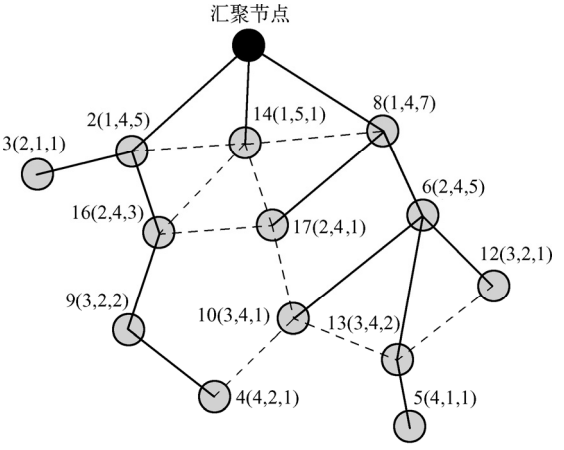


图 3.8　最短路径树

Fig. 3.8　The shortest path tree

在图 3.8 中，8 号节点的数据转发量明显比 14 号节点的大，6 号节点的数据转发量比 17 号节点的大，因此，17 号和 10 号节点需要调整其父节点。经过算法的 Step4 和 Step5，17 号节点和 10 号节点改变自身的父子关系，从而减少了 8 号和 6 号节点的数据转发量，降低了其拥塞程度，最终生成如图 3.9 所示的最短路径负载均衡树（实线表示）。经计算，此时构造的数据汇集树的负载均衡程度约为 3.3262。相比图 3.8 的最短路径树，负载均衡程度得到提高。

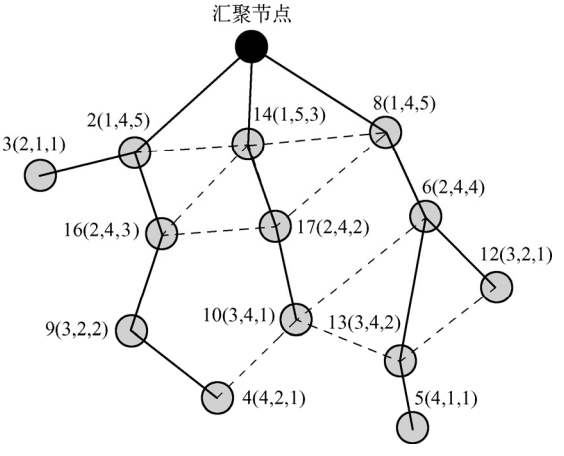


图 3.9　最短路径负载均衡树

Fig. 3.9　Shortest path tree with minimum balance weight

3.3.2　LDGT-SPT 算法理论证明

【定理 3.2】LDGT-SPT 算法收敛。

【证明】显然，LDGT-SPT 算法的 Step0 到 Step3 收敛，而 Step4 经过一个数据采集周期后收敛。Step5 采用反证法证明。假设 Step5 不收敛，存在节点 x，$x\in V$，且存在节点 u 和 v，$u\in F(x)$，$v\in F(x)$，节点 x 不能准确判定 $t_x(u)$与 $t_x(v)$的大小关系。而 LDGT-SPT 算法通过 Notify_Msg 消息、ACK“稍带”、数据“稍带”和“偷听”机制保证了节点 x 可以准确知道 $t_x(u)$与 $t_x(v)$的大小关系，因此矛盾，所以 Step5 收敛。LDGT-SPT 算法的所有步骤收敛，LDGT-SPT 算法收敛得证。

【定理 3.3】LDGT-SPT 算法构造的数据汇集树是一棵最短路径树。

【证明】使用反证法。假设由 $G=(V, E)$，通过 LDGT-SPT 算法构造了树 $T=(V, E')$，$E'\subseteq E$，树 T 不是最短路径树，则存在节点 u，$u\in V$，u 到执行器节点的距离不是最短路径，对于 $\forall v$，$v\in F(u)$，边 $uv\notin E'$。由算法的 Step3 知，必然 $\exists v$，$v\in F(u)$，其边 $uv\in E'$，因此矛盾，定理 3.3 得证。

【定理 3.4】LDGT-SPT 算法构造的数据汇集树的每一层的负载均衡程度最大。

【证明】使用反证法。假设由 $G=(V, E)$，通过 LDGT-SPT 算法构造了树 $T=(V, E')$，$E'\subseteq E$，存在树 T 的第 i 层，$i<N$，其负载均衡程度不是最大，则在第 i 层存在节点 x，y，在第 i+1 层中存在节点 u，$x\in F(u)$，$y\in F(u)$，满足$\{t(u)+t_u(x)<t_u(y)\}$。由算法 Step5 可知，节点 u 的父节点必然发生调整，因此树 T 没有通过 LDGT-SPT 算法完成构造，这与树 $T=(V, E')$是通过 LDGT-SPT 算法构造相矛盾，定理 3.4 得证。

【定理 3.5】LDGT-SPT 算法构造的数据汇集树是一棵最短路径负载均衡树。

【证明】使用反证法。假设由 $G=(V, E)$，通过 LDGT-SPT 算法构造了树 $T=(V, E')$，$E'\subseteq E$，树 T 不是负载均衡树，则由定义 3.2 和定义 3.3 可知，至少存在第 i 层，$i<N$，其负载均衡程度不是最大，但这与定理 3.4 矛盾，定理 3.5 得证。

3.4 仿真验证

3.4.1 仿真环境与参数

仿真实验使用 NS-2 平台，共设置了 100 个节点，其中包括 1 个执行器节点和 99 个传感器节点，在 800m×800m 的场景下随机生成节点的位置，其他仿真参数见表 3.1。传感器节点以一定的周期产生数据并向执行器节点传输，其初始能量为 50J，为了简化计算，定义节点用于感知、接收与传送一个字节数据所耗费的能量分别为 1×10^{-5}J、5×10^{-5}J 和 1×10^{-4}J，执行器节点能量无限制。

表 3.1　仿真参数

Table 3.1　Simulation parameters

参　　数	值
Scene size	800m×800m
Node number	99 nodes + 1 执行器
Mac	802.11
Application	CBR
Packet Size	128
Queue length	10
WirelessPhy:Power for Tx	3.65362-e2(for 100m transmit range)

3.4.2　LDGT-SPT 分组定义

LDGT-SPT 算法主要包括 4 个控制分组，分别为 Request_Msg 分组、Confirm_Msg 分组、Change_Msg 分组和 Notify_Msg 分组，以及两个数据传输分组，分别为 Data 分组和 Ack 分组。

1．Request_Msg 分组

Request_Msg 分组用于子节点通知父节点加入，分组包含的数据域见表 3.2。SrcAddr 表示发出请求消息的源地址，ParentAddr 表示通知加入的父节点，SeqNum 为消息序列号，用于判断新旧消息。T 表示 SrcAddr 节点的 t 值。

表 3.2　Request_Msg 分组数据结构

Table3.2　Structure of Request_Msg

数据域	SrcAddr	ParentAddr	SeqNum	T
长度	16bit	16bit	8bit	8bit

2．Confirm_Msg 分组

Confirm_Msg 分组用于父节点确认子节点加入，分组包含的数据域见表 3.3。SrcAddr 表示发出确认消息的源地址，ChildAddr 表示被确认的子节点，SeqNum 为 Request_Msg 消息的序列号。T 表示 SrcAddr 节点的 t 值。

表 3.3　Confirm_Msg 分组数据结构

Table3.3　Structure of Confirm_Msg

数据域	SrcAddr	ChildAddr	SeqNum	T
长度	16bit	16bit	8bit	8bit

3．Change_Msg 分组

Change_Msg 分组用于子节点通知改变了父节点，分组包含的数据域见表 3.4。

SrcAddr表示发出改变消息的源地址，FromAddr表示脱离的父节点地址，ToAddr表示加入的父节点地址，SeqNum 为 Change_Msg 消息的序列号。T 表示 SrcAddr 节点的 *t* 值。

表 3.4　Change_Msg 分组数据结构

Table3.4　Structure of Change_Msg

数据域	SrcAddr	FromAddr	ToAddr	SeqNum	T
长度	16bit	16bit	16bit	8bit	8bit

4．Notify_Msg 分组

Notify_Msg 分组用于节点通知上层节点 *t* 值的变化，分组包含的数据域见表 3.5。SrcAddr 表示发出通告消息的源地址，ParentAddr 表示通告的父节点，SeqNum 为消息序列号，用于判断新旧消息。T 表示 SrcAddr 节点的 *t* 值。

表 3.5　Notify_Msg 分组数据结构

Table3.5　Structure of Notify_Msg

数据域	SrcAddr	ParentAddr	SeqNum	T
长度	16bit	16bit	8bit	8bit

5．Data 分组

Data 分组用于数据传输，分组包含的数据域见表 3.6。SrcAddr 表示数据产生的源地址，ParentAddr 表示下跳父节点，SeqNum 为消息序列号，用于 ACK 确认，Len 表示 Data 的数据长度，Data 表示传感器数据。

表 3.6　Data 分组数据结构

Table3.6　Structure of Data

数据域	SrcAddr	ParentAddr	SeqNum	Len	Data
长度	16bit	16bit	16bit	8bit	8*Len bit

6．Ack 分组

Ack 分组用于数据传输，分组包含的数据域见表 3.7。SrcAddr 表示数据的源地址，ChildAddr 表示确认的子节点地址，SeqNum 为消息序列号，与 Data 分组匹配，Flag/T 表示接收状态或节点的 *t* 值。

表 3.7　Ack 分组数据结构

Table3.7　Structure of Ack

数据域	SrcAddr	ChildAddr	SeqNum	Flag/T
长度	16bit	16bit	16bit	8bit

3.4.3　仿真结果

本书在不同的数据发送速率下，对网络吞吐量、丢包率、延时和网络寿命进行评估，以及对负载均衡因子进行了比较。仿真试验对比了普通的最短路径树（SPT）、静态负载平衡树方法（Static Load-Balanced Tree，SLBT）和 AODV 路由协议[125]，其中，SLBT 方法将网络节点构成一棵负载平衡树，数据通过平衡树传输到执行器节点。构造平衡树的算法使用了文献[97]提出的方法。

1．网络吞吐量

网络吞吐量是网络层的一个重要性能指标，通常是指点对点的数据流量测试。但在无线传感器网络数据汇集应用中，每个节点都要产生数据并向执行器节点传输。因此，在仿真实验中，统计单位时间内执行器节点接收到 CBR 数据包的个数，得到与网络吞吐量相当的一个性能指标。图 3.10 显示了各种算法的网络吞吐量，LDGT-SPT 的网络吞吐量大于 SPT、SLBT 和 AODV。经计算，在数据发送速率较高（数据包产生速率大于 1 包/s）时，相比 SPT 和 SLBT，LDGT-SPT 网络吞吐量提高了分别约 18%和约 15%，说明 LDGT-SPT 生成的数据汇集树的负载均衡效果更好，使得网络吞吐量有明显提高。

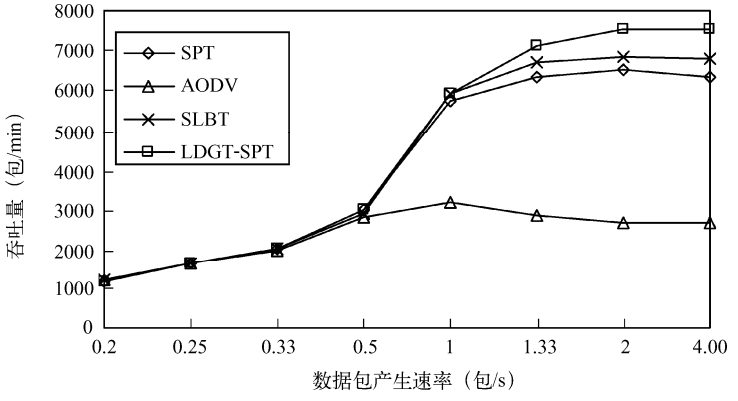


图 3.10　吞吐量比较

Fig. 3.10　Compare of the network throughput

2．丢包率

在仿真实验中，统计单位时间内各个传感节点的应用层发出的 CBR 包个数和执行器节点接收到 CBR 包的个数，用总的发送数据包个数减去总的接收数据包个数后，除以总的发送数据包个数就可以得到丢包率。图 3.11 给出了传感器节点在不同数据包产生速率下的丢包率。随着发送速率的增大，各个算法的丢包

率都有所增加，但 LDGT-SPT 的丢包率增幅明显小于 SPT、SLBT 和 AODV。经计算，在数据发送速率较高时（数据包产生速率大于 1 包/s），相比 SPT 和 SLBT，LDGT-SPT 的丢包率减少了约 28%和约 15%，说明 LDGT-SPT 生成的数据汇集树，在很大程度上防止了由于热节点导致的丢包，并且减少相邻路径的串扰。

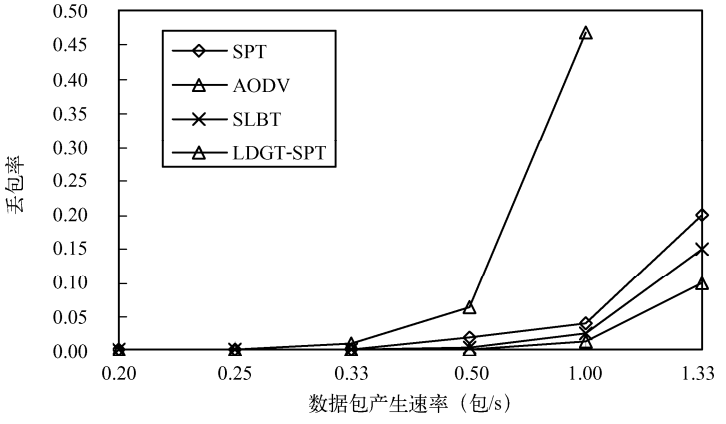


图 3.11　丢包率比较

Fig. 3.11　Compare of the packet loss rate

从网络吞吐量和丢包率的仿真结果可见，当数据发送速率较高（数据包产生速率大于 0.5 包/s）时，网络就会发生一定程度的拥塞，AODV 协议的丢包率急剧增大，网络吞吐量不再增加。因此，AODV 协议不适用于本书所述的网络应用环境（无线传感器网络数据汇集应用），在网络延时分析中不再与 AODV 协议进行对比。

3．网络延时

在仿真过程中，选取一个离执行器节点较远的传感器节点作为测试节点，并统计其产生的数据包传输到执行器节点的时间。如图 3.12 所示，在拥塞情况比较明显时（数据产生速率大于 1），数据延时明显增加。这是由于数据流量较大时，冲突和重传次数增加，使得发送缓存队列增长，从而产生了较大的排队延时。但 LDGT-SPT 的流量均衡策略使得分组延时小于普通的 SPT。SLBT 没有采用最短路径树，因此，相比 SPT 和 LDGT-SPT，其平均延时最大。经计算，LDGT-SPT 比 SLBT 的平均延时减少约 360%。在数据发送速率较高（数据包产生速率大于 1 包/s）时，相比 SPT，LDGT-SPT 网络延时减少了约 30%。

4．负载均衡因子

在无线传感器网络数据汇集应用中，数据从距离执行器节点远端向执行器节点汇集，因此执行器邻居节点的负载最重，执行器邻居节点的寿命长短往往

决定了网络寿命的长短。仿真试验中，将节点的数据转发量作为节点负载，并对执行器邻居节点的负载均衡因子进行了计算。结果如图 3.13 所示，SPT、SLBT 和 LDGT-SPT 的负载均衡因子分别为 0.35、0.93 和 0.89。LDGT-SPT 与 SPT 相比，负载均衡因子提高了约 166%，而与 SLBT 相当（差值小于 5%）。

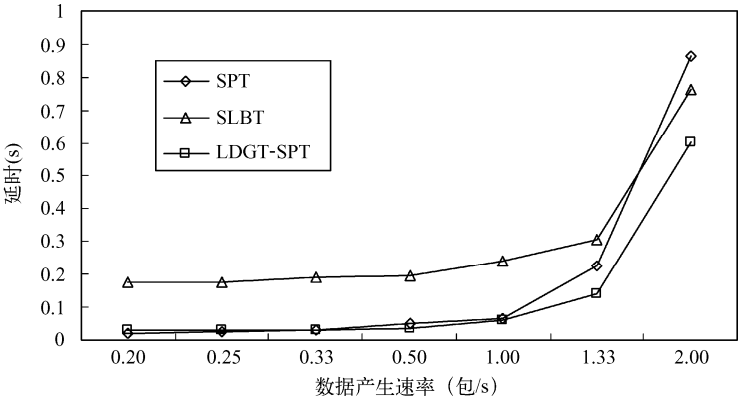


图 3.12　网络延时比较

Fig. 3.12　Compare of packet delay

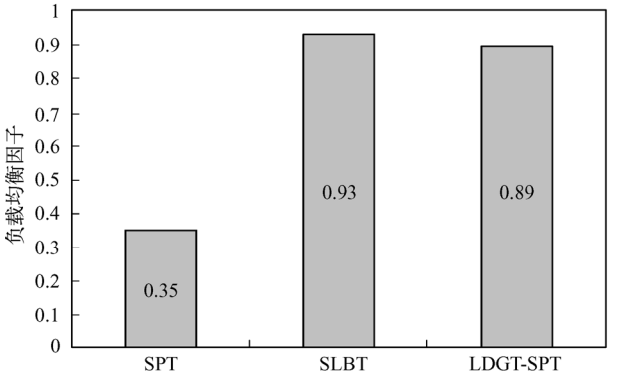


图 3.13　负载均衡因子比较

Fig. 3.13　Compare of load-balance factor

5．网络寿命

本次仿真实验将网络寿命定义为：从网络开始工作到第 1 个传感器节点消耗完自身能量所经历的时间间隔。网络寿命情况如图 3.14 所示，由于 LDGT-SPT 均衡了网络节点的数据流量，减少了热节点的数据转发量，因此，网络寿命较普通的 SPT 更长。经计算，LDGT-SPT 相比 SPT 网络平均寿命延长了约 40%。LDGT-SPT 的网络寿命与 SLBT 相当（平均差值小于 5%）。

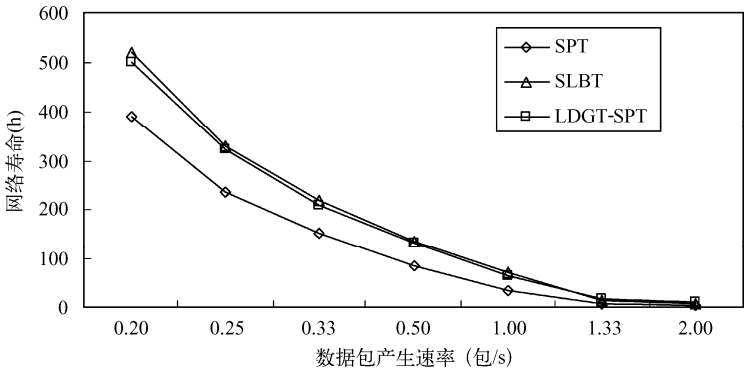


图 3.14　网络寿命比较

Fig. 3.14　Compare of network lifetime

综上所述，LDGT-SPT 相比 SPT 在各个性能指标上都有明显改善，相比 SLBT 虽然在负载均衡程度和寿命上相当，但在网络吞吐量、丢包率和延时方面有明显优势。相比 SLBT，LDGT-SPT 无须获取全局拓扑信息，算法更加实用。

3.5　小结

本章针对无线传感器网络数据汇集应用，节点固定且数据产生的速率一致时，通过构建负载均衡数据汇集树的方式均衡网络负载，在最短路径树拓扑结构保证数据实时性的情况下，建立了最短路径负载均衡树。提出的无线传感器网络负载均衡数据汇集树建立算法（LDGT-SPT）使得网络数据流均衡分配，从而缓解了网络拥塞，提高了网络性能。理论分析证明了 LDGT-SPT 算法收敛，能构造一棵最短路径负载均衡树。仿真实验表明，本书提出的 LDGT-SPT 算法虽然在负载均衡程度和网络寿命上与相关算法相当，但是在网络性能方面有显著的提高。

参 考 文 献

[1] Tilak S, Abu-Ghazaleh N B, Heinzelman W. A Taxonomy of Wireless Micro- Sensor Network Models[J].

[2] Qiu W, Skafidas E, Hao P. Enhanced tree routing for wireless sensor networks[J]. Ad Hoc Networks. 2008, 7(3): 638-650.

[3] Perkins C, Royer E M, Das S R. Ad hoc On-Demand Distance Vector (AODV) Routing[J]. Internet RFCs. 2003.

第 4 章　基于 ACO 的动态负载均衡数据汇集算法

在无线传感器网络数据汇集应用中，网络数据流量相对于其他类型（查询驱动型和事件驱动型）大。如果各个数据流分支的流量分配不均衡，则会导致部分上游节点负载过重，能量快速耗尽，造成节点提前“死亡”，缩短网络寿命。

第 3 章从静态负载均衡角度研究了构造负载均衡数据汇集树的方法，但当数据源产生数据速率不一致（例如：一个应用中采用了多种传感器，每种传感器有不同的数据采集周期）时，通过构造负载均衡数据汇集树的方法就不能够达到负载均衡的效果，这时需要采用动态负载均衡的方法。本章基于蚁群优化（Ant Colony Optimization，ACO）[1]的思想，提出一种动态负载均衡的无线传感器网络数据汇集算法（Load-balanced Data Gathering algorithm based on ACO，LDG-ACO）。LDG-ACO 算法作用于层次网络结构，减少了路径搜索的盲目性。LDG-ACO 算法根据不同的任务，将蚂蚁分为前向探索蚂蚁、前向运输蚂蚁和后向蚂蚁三类，采用节点的负载信息作为启发因子，使得蚂蚁具有负载感知的功能，并趋向于走负载低的路径。在数据汇集过程中，前向探索蚂蚁充当路径搜索者的功能，并和前向运输蚂蚁承担数据搬运的功能，其转移概率按照信息素少概率大的原则计算。后向蚂蚁用于整条路径的信息素更新，通过向路径洒信息素的方式，将上游节点的负载信息反馈给下游的前向蚂蚁，使得前向蚂蚁在路径选择时，具有更多的路径负载信息。前向蚂蚁与后向蚂蚁的任务协作充分体现了群体智能的社会分工合作能力。4.4 节的仿真实验表明，与相关算法相比，LDG-ACO 算法在网络性能、负载均衡程度和网络寿命上都有显著提高。

4.1　ACO 的优点与不足

ACO 是由意大利学者 Dorigo M.等人通过模拟自然界中蚂蚁集体觅食行为提出的一种利用正反馈原理解决全局优化问题的启发式算法。蚁群中每个个体的能力有限，行为规则简单，只能感知局部信息，不能获得全局信息。无线传感器网络中单个节点的计算能力、存储器空间、能量和通信半径都有限，使得每个节点只能感知邻居节点，并与之发生信息交互。无线传感器网络中的单个

节点与 ACO 中的个体非常相似。因此，采用 ACO 的思想来解决无线传感器网络中的相关问题是非常契合的。ACO 采用分布式控制，不存在直接的控制中心，算法具有潜在的并行性，路径搜索可以同时从多个点进行，非常符合无线传感器网络的相关应用场景。ACO 已在无线传感器网络的相关应用中取得了较好的效果[2~9]。

虽然 ACO 在无线传感器网络的一些领域取得了较好的效果，但在无线传感器网络的数据汇集应用方面有不合理之处，存在一定的局限性。由于 ACO 本身所具有的收敛性质，会使得大量蚂蚁汇聚到少数路径上，这些路径上的节点能量将急剧消耗，导致节点提前“死亡”，缩短整个网络的生命周期，严重时会造成这些路径上出现拥塞，最终导致整个网络瘫痪。这种现象对于要求负载均衡的无线传感器网络数据汇集应用是不合理的。如图 4.1 所示，蚂蚁从食物源 F 到蚁巢 H 的过程中，存在三条路径。根据 ACO 的收敛性质，必然导致绝大多数蚂蚁汇聚到其中的某条路径上（假设为 $F \rightarrow B \rightarrow H$）。如果 F 为一个网络，其数据流量巨大，极易造成转发节点 B 的负载过重，严重时会发生拥塞，如果没有其他路由修复机制，则会使网络瘫痪。即使不发生拥塞，节点 B 的能量快速消耗，会导致其提前“死亡”，缩短整个网络的生命周期。传统的 ACO 在完成路径搜索后就终止，网络陷入瘫痪的可能性更大。无线传感器网络数据汇集应用中的数据源在网络的整个生命周期有效，网络拓扑或者路径信息一旦发生变化，传统的 ACO 需要重新搜索路径，这将带来很大的开销。

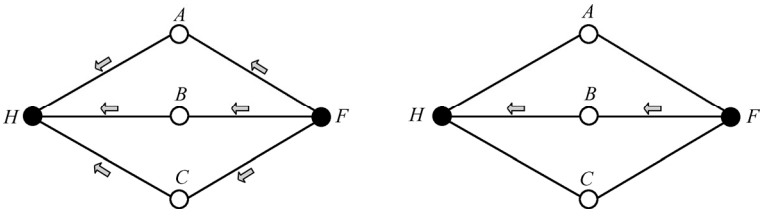


图 4.1　ACO 的局限性

Fig. 4.1　Limitation of ACO

综上所述，ACO 的个体特征与无线传感器网络的节点情况非常相似，其算法思路适合解决无线传感器网络的相关问题，但针对无线传感器网络数据汇集应用，ACO 存在改进的必要。

4.2　LDG-ACO 算法原理

由 4.1 节可知，在无线传感器网络数据汇集应用中，由于 ACO 的收敛性质

导致大量数据聚集到少量路径上传输，使得这些路径上的节点能量消耗过快，从而影响了网络寿命。对于一个如图 4.2 所示的无线传感器层次网络，数据总是从下游向上游传输（子节点将数据传输给父节点），最终都可以到达执行器节点。对于层次网络中的任何一个节点，如果它知道其每个父节点的负载情况，甚至父节点到执行器节点路径上所有节点的负载情况，就可以根据上游的负载情况，将数据转发给当前最小负载路径的父节点，从而最大限度地均衡父节点到执行器节点所有路径上节点的负载。负载信息可以用信息素和剩余能量表示。LDG-ACO 算法将节点的每次数据传输（或转发）过程看作蚂蚁搬运数据的一个步骤。蚂蚁每经过一个节点，都洒下一定量的信息素，同时，环境在一定程度上使信息素消失。因此，路径上信息素的多少表示当前路径负载的大小，信息素越多，表示当前路径上节点转发的数据量越大，负载相对较重。如果蚂蚁每次走信息素少的路径，则节点的每次数据传输（或转发）过程沿着负载低的路径进行。从整体上看，各个层次的节点负载呈现出均衡分配。

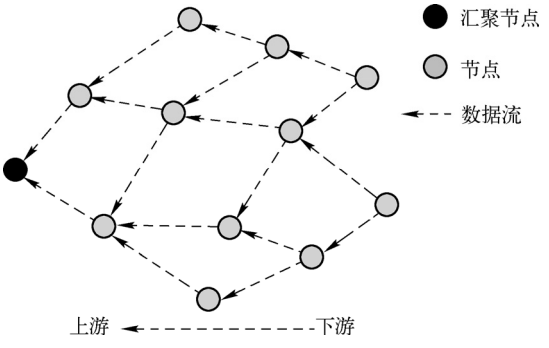


图 4.2　无线传感器层次网络

Fig. 4.2　Wireless sensor level structure network

4.3　LDG-ACO 算法描述

4.3.1　LDG-ACO 算法术语与规则

对于一个数据汇集的无线传感器层次网络，数据流的方向总是从低层次的数据源节点指向高层次的执行器节点。可以通过第 3 章提出的动态交叉退避窗口算法进行层次发现，得到节点的层次关系和父子关系。

LDG-ACO 算法作用于已建立的层次网络结构上，并根据不同的任务将蚂蚁分为三类：

（1）前向探索蚂蚁（FD-ANT）：从食物源（数据源节点）产生的蚂蚁，用于路径探索，兼顾数据运输。在行走过程中，FD-ANT 需要记住自身经过的路径，以便按原路径返回。

（2）前向运输蚂蚁（FT-ANT）：从食物源（数据源节点）产生的蚂蚁，仅用于数据运输。在行走过程中，FT-ANT 无须记住自身经过的路径，相对于 FD-ANT 开销小。

（3）后向蚂蚁（B-ANT）：从蚁巢（执行器节点）返回食物源（数据源节点）的蚂蚁，用于路径拥塞程度的检测和通告。

FD-ANT 和 FT-ANT 统称为前向蚂蚁，FD-ANT 到达蚁巢（执行器节点）后转化为 B-ANT。

【规则 4.1】（前向蚂蚁产生规则）：由于 FD-ANT 需要记住路径，会产生一些额外的开销，因此，FD-ANT 仅在前向蚂蚁中占一定的比例。FD-ANT 按照式（4.1）产生，FT-ANT 按照式（4.2）产生。

FD-ANT 的产生速率：

$$M(i) = k \cdot f(i) \tag{4.1}$$

FT-ANT 的产生速率：

$$N(i) = (1-k) \cdot f(i) \tag{4.2}$$

式（4.1）和式（4.2）中，$k \in (0,1]$为比例因子；$f(i)$为节点 i 产生数据包的速率（包/min）。

无线传感器网络节点的负载量化是一个比较复杂的问题，但是可以从两个方面来估计：一方面，网络寿命最大化的关键在于减少节点能耗，延长节点的生存周期，因此，剩余电量可以一定程度地表征节点的负载情况（如节点剩余电量少，可以认为其处理的任务较多、负载较重。因此，剩余电量和负载成反比关系）；另一方面，节点消息转发量的大小与网络性能有密切关系（如丢包率、网络时延以及网络拥塞程度等），因此，消息转发量也是衡量节点负载程度的一个重要指标。消息转发量可以用处于消息缓冲队列中的待处理消息数来表示（如消息缓冲队列越长，则需要转发消息的任务越重。因此，消息缓冲队列长度和负载成反比关系）。综上所述，有如下定义：

【定义 4.1】无线传感器网络节点负载定义：

$$L(i) = \frac{1}{\lambda_e \dfrac{e(i)}{E(i)} + \lambda_q \dfrac{q(i)}{Q(i)}} \tag{4.3}$$

式中，$L(i)$ 为节点 i 的负载；$q(i)$ 为节点 i 的空闲队列长度；$Q(i)$为节点总队列

长度；$e(i)$为节点 i 的剩余能量；$E(i)$为节点 i 的总能量或称初始能量；λ_e 和λ_q分别为能量和队列的权重。消息在队列中会产生排队延时，因此，能量和队列的权重由实际应用对实时性的要求确定。

【定义 4.2】LDG-ACO 的启发因子 η_{ij} 定义为

$$\eta_{ij}=\frac{1}{L(j)} \tag{4.4}$$

式中，$L(j)$为节点 i 的下跳节点 j 的负载。启发因子 η_{ij} 表示蚂蚁从节点 i 转移到节点 j 的期望程度，使得前向蚂蚁趋向于走负载低的路径。

【规则 4.2（LDG-ACO 的转移概率）】

$$P_{ij}(k)=\begin{cases}\dfrac{\tau_{ij}^{-\alpha}\eta_{ij}^{\beta}}{\sum\limits_{s\in F(i)}\tau_{is}^{-\alpha}\eta_{is}^{\beta}}, j\in F(i)\\ 0, j\notin F(i)\end{cases} \tag{4.5}$$

式中，$P_{ij}(k)$为第 k 只前向蚂蚁从节点 i 转移到节点 j 的概率；τ_{ij} 表示边(i,j)上的信息素量；η_{ij} 为启发因子；α 和 β 分别表示信息素和启发式因子的相对重要程度；$F(i)$为节点 i 的父节点。

LDG-ACO 的转移概率与标准 ACO 的转移概率[126]相比，有三个不同点：①系统参数α负变换，这样使得前向蚂蚁按照信息素少优先的原则选择路径，将标准 ACO 的收敛性改为发散性；②下跳节点的可行域从邻节点域变成父节点域，减小了前向蚂蚁搜索路径的盲目性，并且路径不会出现回环，无须存储禁忌表，减少了算法对存储器的开销；③标准 ACO 通常定义启发因子 η_{ij} 为距目的地的距离，而 LDG-ACO 算法作用于层次网络，下跳节点到达执行器节点的距离相等，启发因子 η_{ij} 定义为下跳节点的负载倒数，更能体现负载均衡的思想。

【规则 4.3（前向蚂蚁对环境信息素的更新规则）】如果路径(i,j)是前向蚂蚁 k 走过的路径，则按照式（4.6）进行信息素的局部更新，否则按照式（4.7）更新。

$$\tau'_{ij}=(1-\rho)\tau_{ij}+\sum_{k=1}^{m}\Delta\tau_{ij}^{k} \tag{4.6}$$

$$\tau'_{ij}=(1-\rho)\tau_{ij} \tag{4.7}$$

式（4.6）和式（4.7）中，$\rho\in[0,1)$，表示环境对信息素的挥发系数；式（4.6）中，$\Delta\tau_{ij}{}^{k}$ 为蚂蚁 k 在路径(i,j)上洒下的信息素量。

前向蚂蚁按照信息素少优先的原则选择路径，使父节点的负载趋向于均衡，但前向蚂蚁对环境信息素的更新并不能使得上游链路的负载情况及时地反映给

下游节点。如图 4.3 所示，食物源 F_2 的出现，增加了(A, H)上的蚂蚁数量，但并不影响从食物源 F_1 出发的蚂蚁对(F_1, B)和(F_1, D)的路径转移概率，从 F_1 出发的两条路径上的蚂蚁数量几乎相当。设从食物源 F_1 和 F_2 出发的蚂蚁个数分别为 m 和 n，则经过 A 的蚂蚁数量为 $m/2+n$，而经过 C 节点的蚂蚁数量为 $m/2$，显然，节点 A 的负载比节点 C 的负载重。因此，需要在后向蚂蚁从蚁巢回到食物源的过程中对路径上的信息素进行一次全局更新。

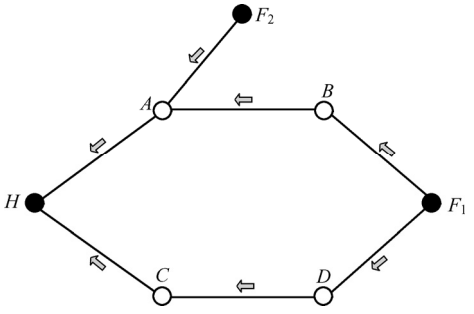


图 4.3　前向蚂蚁对环境信息素的更新

Fig. 4.3　Update pheromone by forward ants

【规则 4.4（后向蚂蚁对环境信息素的更新规则）】如果路径(i, j)是后向蚂蚁走过的路径，则按照式（4.8）进行信息素的局部更新，否则不更新。

$$\tau'_{ij} = \tau_{ij} + \frac{\max[\tau_{\text{path}(i,j)}]}{2} \tag{4.8}$$

式中，$\max[\tau_{\text{path}(ij)}]$表示从食物源节点到蚁巢节点经过$(i, j)$的路径上最大的信息素。

后向蚂蚁在对环境信息的更新过程中，将路径上的最大负载情况通告到下游节点的路径上，这就为前向蚂蚁感知整条路径的负载情况提供了参考信息。

4.3.2　LDG-ACO 算法步骤

Step1：通过动态交叉退避窗口算法进行层次发现，得到节点的层次关系以及父子关系。

Step2：初始化各条父子链路的信息素、节点的剩余能量和负载。

Step3：数据源节点按照规则 4.1 产生前向蚂蚁。

Step4：每只前向蚂蚁按照规则 4.2 选择下一跳节点，并且按照规则 4.3 对环境信息素进行更新。

Step5：如果前向蚂蚁到达蚁巢（执行器节点），则前向蚂蚁中的 FD-ANT 变为后向蚂蚁并按原路径返回，并按照规则 4.4 对环境信息素进行更新，否则

返回 Step4。

Step6：如果后向蚂蚁返回食物源（数据源节点），则蚂蚁使命完成（死亡），否则继续按照规则 4.4 对环境信息素进行更新。

Step7：返回 Step3。

在 LDG-ACO 算法的整个步骤中，前向蚂蚁需要负责数据传输，为了保证数据传输的可靠性，通常需采用 ACK 确认。因此，父节点的剩余能量和信息素可以通过 ACK“稍带”的方式告知子节点，以减少数据传输的额外开销。LDG-ACO 算法的核心代码（主程序伪代码、Timer 中断伪代码和接收消息伪代码）分别如下。

LDG-ACO 算法主程序伪代码：

```
/*LDG-ACO 算法主流程部分*/
#define ALPHA 1
#define BETA 0.5
#define LAMBDA 1
#define PHI 0.5                  //信息素挥发系数
#define UPDATE_TIME 100          //ms，环境信息素更新时间间隔
#define MIN_PH 1                 //最小信息素
#define MAX_PH 1024              //最大信息素
#define DERTA_PH 10              //每只蚂蚁在转移时洒向路径的信息素量
#define K 0.2                    //percent(%) 比例因子
Main() {
  FartherList=NULL，state=HELLO_PENDING
  switch(state) {
  case HELLO_PENDING:
    SendHello();
    /*在消息接收中断中更新 NeighborNum */
    HelloTimerStart();
    /*在定时器中断中更改 state=LAYER_FINDING */
    break;
  case LAYER_FINDING:
    DOBW(); //执行动态交叉退避窗口算法
    state=DATA_GATHERING;
    break;
  case DATA_GATHERING:
    TimerStart(T, PERIOD_MODEL); //启动周期性定时器，数据采集周期为 T
    /*在定时器中断中采集并传输数据*/
    break;
  }
}
```

LDG-ACO 算法 Timer 中断伪代码：

```
/*LDG-ACO 算法 Timer 中断部分*/
TimerHandle() {
  switch(state) {
  case HELLO_PENDING:
    SendHello();
    if (HelloCount < HELLO_TIMES) HelloTimerStart();
    else {
      state=LAYER_FINDING
      if(LOCAL_ADDR==执行器) SendLayerMsg();//由执行器节点开始层次发现
    }
    break;
  case DATA_GATHERING:
    if(isInit)Init FartherList();          //初始化父子链路的信息素和负载
    TimerStart(T, PERIOD_MODEL);           //启动周期性定时器，数据采集周期为 T
    SensorStart();                         //采集数据
    if(GetPro < K)                         /概率小于比例因子，则产生 FD-ANT
      MakeFDANTPkt();                      //使用 FD-ANT 运输数据
    else
      MakeFTANTPkt();                      //使用 FT-ANT 运输数据
    SendMsg();                             //发送数据
    break;
  }
}
```

LDG-ACO 算法接收消息伪代码：

```
/*LDG-ACO 消息接收部分*/
MsgHandle(msg){
  switch(msg->type) {
    case HELLO:
    UpdateNeighbor (msg);                  //更新邻居节点信息
    break;
  case Layer_Msg:
    UpdateFartherList(msg);                //更新父子链路
    forward(msg);                          //转发数据
    break;
  case FD-ANT:
    if (LOCAL_ADDR == 执行器)
      MakeBANTPkt(msg);                    //产生 B-ANT
    else {
      UpdatePh();                          //更新环境信息素
```

```
            UpdateFDANTPkt(msg);        //更新 F-DANT
          }
          forward(msg);                 //转发数据给下一个节点或应用端口
          break;
        case FT-ANT::
          UpdatePh(msg);                //更新环境信息素
          forward(msg);                 //转发数据给下一个节点或应用端口
          break;
            case B-ANT:
          UpdatePh(msg);                //更新环境信息素
          UpdateBANTPkt(msg);           //更新 F-DANT
          forward(msg);                 //转发数据给下一个节点
          break;
            case ACK:
          UpdateParentLoad (msg);       //更新父节点的负载信息
         break;
       }
    }
```

4.4　仿真验证

4.4.1　仿真环境与参数

由于 TOSSIM 平台对仿真环境全局管理很方便，可以很方便实现环境对节点信息素的实时更新，因此，本次仿真实验使用 TOSSIM 平台。设置 100 个网络节点，随机部署在仿真区域，0 号节点为执行器节点。Radio model 采用 Fixed radius(10.0)，Distance scaling factor 设置为 0.8，形成网络拓扑结构如图 4.4 所示。设置传感器节点的初始能量为 50J，为了简化计算，定义节点用于感知、接收与传送一个字节数据所耗费的能量分别为 1×10^{-5}J、5×10^{-5}J 和 1×10^{-4}J，执行器节点能量无限制。设置仿真次数为 6 次。随机选择部分节点作为数据源节点、为了使不同的数据源节点具有不同的数据采集周期，在六次实验中，数据源节点在启动时分别在(0s-1s)、[0.5s-1.5s)、[1s-3s)、[3s-7s)、[5s-15s)、[15s-25s)、[25s-35s)、[35s-45s)、[45s-55s)、[55s-65s)时间段内随机获得数据采集周期。LDG-ACO 算法中，根据经验设置参数 α=1，β=0.5，ρ=0.5，$\Delta\tau_{ij}$=10，λ_e= λ_q=1，环境每隔 100ms 对节点的信息素进行一次更新，为了保证计算结果不溢出，设置每个节点的初始信息素和最小信息素 MIN_PH=1.0。

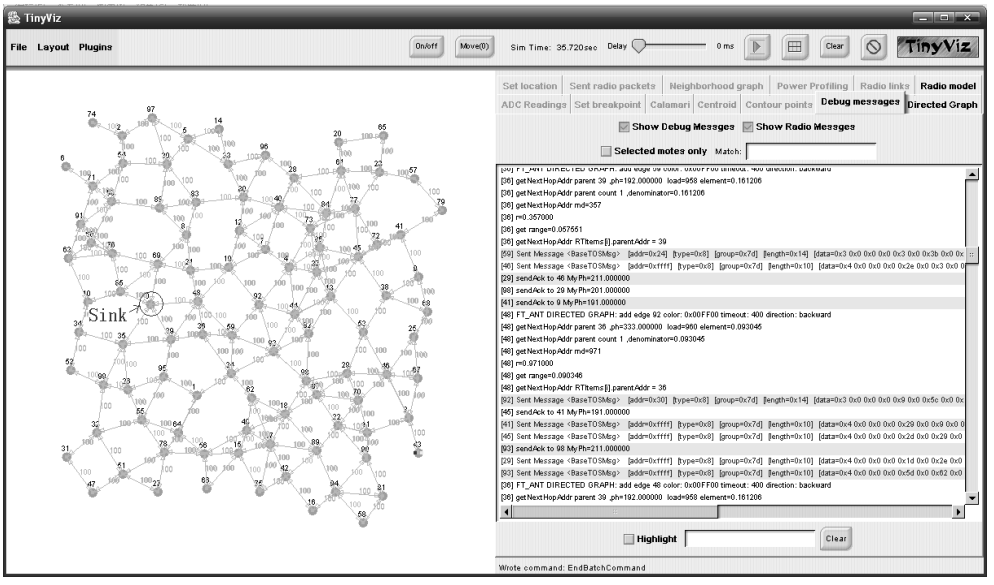


图 4.4　仿真实验环境与节点部署

Fig. 4.4　Simulation environment and deployment of nodes

4.4.2　LDG-ACO 分组定义

LDG-ACO 算法主要包括四种分组，分别为 FD_ANT 分组、FT_ANT 分组、ACK 分组和 B_ANT 分组。

1．FD_ANT 分组

FD_ANT 分组用于数据传输和路径探索，由数据源节点发起。FD_ANT 分组包含的数据域见表 4.1，SrcAddr 表示数据源地址，PrevHop 表示转发节点的地址（当数据源发出该消息时，PrevHop 与 SrcAddr 一致），NextHop 表示下跳节点的地址，SeqNum 为序列号，DataLen 表示数据域 Data 的长度（字节数），HopCount 表示地址链表域中的地址个数，Data 为数据域，AddrChain 为地址链表域。

表 4.1　FD_ANT 分组数据结构

Table 4.1　Structure of FD_ANT

数据域	SrcAddr	PrevHop	NextHop	SeqNum	DataLen	HopCount	Data	AddrChain
长度	16bit	16bit	16bit	8bit	8bit	8bit	8*DataLen bit	16*HopCount bit

2．FT_ANT 分组

FT_ANT 分组用于数据传输，由数据源节点发起。FT_ANT 分组包含的数

据域见表 4.2，SrcAddr 表示数据源地址，PrevHop 表示转发节点的地址（当数据源发出该消息时，PrevHop 与 SrcAddr 一致），NextHop 表示下跳节点的地址，SeqNum 为序列号，DataLen 指示数据域 Data 的长度（字节数），Data 为数据域。

表 4.2　FT_ANT 分组数据结构

Table 4.2　Structure of FT _ANT

数据域	SrcAddr	PrevHop	NextHop	SeqNum	DataLen	Data
长度	16bit	16bit	16bit	8bit	8bit	8*dataLen bit

3．ACK 分组

ACK 分组用于确认 FD_ANT 分组和 FT_ANT 分组的数据传输，以及节点信息素和负载的通告。ACK 分组包含的数据域见表 4.3，SrcAddr 表示 ACK 源地址，DstAddr 为 ACK 目的地址，SeqNum 为序列号（前向蚂蚁的序列号），Load 为负载信息，Pheromone 为信息素量。

表 4.3　ACK 分组数据结构

Table 4.3　Structure of ACK

数据域	SrcAddr	DstAddr	SeqNum	Load	Pheromone
长度	16bit	16bit	8bit	16bit	16bit

4．B_ANT 分组

当执行器节点收到一个 FD_ANT 分组后，就产生一个 B_ANT 分组，用于路径拥塞程度检测和通告。B_ANT 分组包含的数据域见表 4.4，DstAddr 为数据源地址（产生 FD_ANT 的地址），PrevHop 表示父节点的地址（当执行器节点发出该消息时，PrevHop 与 SrcAddr 一致），NextHop 表示下跳节点的地址，SeqNum 为序列号（FD_ANT 的序列号），Pheromone 为信息素量，HopCount 表示地址链表域中地址的个数，AddrChain 为地址链表域。

表 4.4　B_ANT 分组数据结构

Table 4.4　Structure of B _ANT

数据域	DstAddr	PrevHop	NextHop	SeqNum	Pheromone	HopCount	AddrChain
长度	16bit	16bit	16bit	8bit	16bit	8bit	16*hopCount bit

4.4.3　仿真结果

定义网络寿命为从网络开始工作到第 1 个传感器节点消耗完自身能量所经历的时间间隔。时间间隔越长，网络寿命越长。

在前向蚂蚁产生的规则中，比例因子 k 控制 FD-ANT 产生的速率。由于 FD-ANT 需要存储路径信息，会增加额外开销，从开销方面考虑，k 值越小越好。FD-ANT 到达蚁巢（执行器节点）后变为 B-ANT。B-ANT 的主要任务是将上游路径的拥塞程度反馈给下游节点，为前向蚂蚁感知整条路径的负载情况提供参考信息。拥塞程度信息反馈越及时，就需要越多的 FD-ANT，从负载均衡方面考虑，k 值越大越好。因此，比例因子 k 的选择需要从额外开销和反馈实时性两个方面综合考虑。表 4.5 给出平均数据采集周期为 60s 时，k 在不同取值下的负载均衡因子和网络寿命情况。可见，在本次仿真环境下，k 取 0.2 使得网络寿命最长。

表 4.5　比例因子 k 对网络性能的影响

Table 4.5　Impact of k

比 例 因 子	k=0.1	k=0.2	k=0.4	k=0.6	k=0.8
平均负载平衡因子 θ	0.85	0.88	0.90	0.91	0.912
网络寿命（h）	1086	1194	1178	1028	897

本次仿真实验对比了五种方法：①最短路径树方法 SPT（the Shortest Path Tree）。在转发数据时，SPT 算法随机选择距离执行器节点最近的节点作为转发节点。②静态负载平衡树方法 SLBT（Static Load-Balanced Tree），网络节点构成一棵负载平衡树，数据通过平衡树传输到执行器节点。构造平衡树的算法使用了文献[97]提出的方法，静态平衡树的结构在数据汇集过程中不发生改变。③动态负载平衡树方法 DLBT（Dynamic Load-Balanced Tree），DLBT 算法基于文献[134]提出的方法，节点使用动态负载平衡树将数据包发送到执行器节点，在数据汇集过程中，节点之间交换剩余能量信息，并根据剩余能量情况动态调整树的结构。④基于标准 ACO 的路由算法（α=0.5，β=2）。⑤本书提出的动态负载均衡方法 LDG-ACO。

1．网络吞吐量

网络吞吐量性能比较如图 4.5 所示，标准 ACO 的吞吐量最差，其次是 SPT，本书提出的 LDG-ACO 算法的吞吐量最大，其次是 DLBT 和 SLBT。经计算，相比 SLBT 和 DLBT，LDG-ACO 的吞吐量分别提高了约 13%和约 30%，说明本书提出的 LDG-ACO 的负载均衡效果更好，使得网络吞吐量有明显提高。

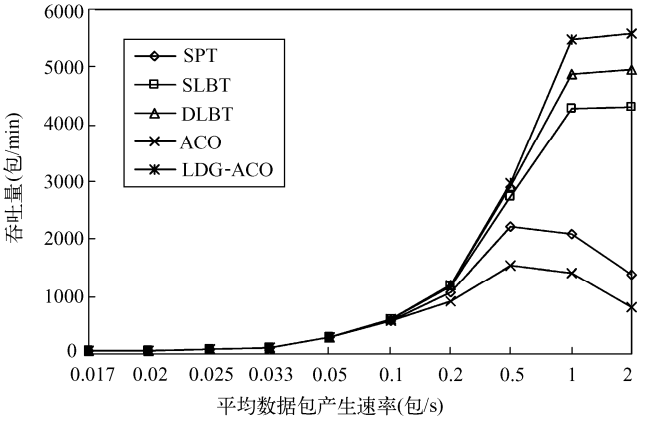


图 4.5　吞吐量比较

Fig. 4.5　Compare of network throughput

2．丢包率

网络丢包率如图 4.6 所示，本书提出的 LDG-ACO 算法的丢包率最小。随着发送速率的增大，各个算法的丢包率都有所增加，但 LDG-ACO 算法的丢包率增幅明显小于 ACO、SPT、SLBT 和 DLBT 的。其中，ACO 和 PT 的丢包率在发送速率达到 0.2（packet/s）时已经很大（分别是约 0.28 和约 0.1）。经计算，相比 SLBT 和 DLBT，LDG-ACO 的平均丢包率分别降低了约 63%和约 52%，说明本书提出的 LDG-ACO 算法的负载均衡效果更好，使得丢包率有明显下降。

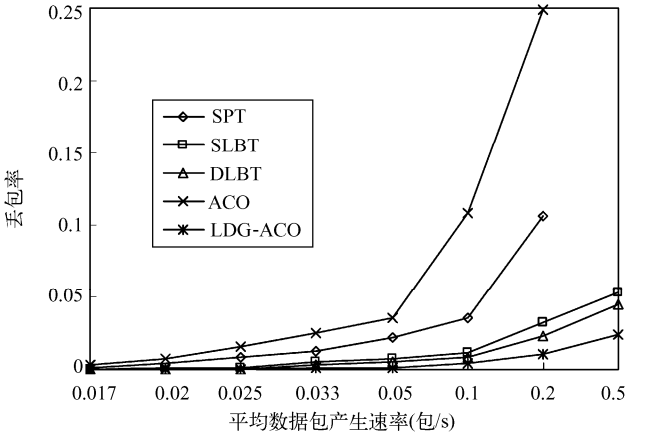


图 4.6　丢包率比较

Fig. 4.6　Compare of packet loss rate

3．网络延时

在网络延时方面，选取一个离执行器节点较远的传感器节点作为测试节点。

结果如图 4.7 所示。本书提出的 LDG-ACO 算法的网络延时最小。随着发送速率的增大，各个算法的网络延时都有所增加，但 LDG-ACO 算法的网络延时增幅明显小于 ACO、SPT、SLBT 和 DLBT 的。其中，ACO 和 PT 的网络延时增幅最大。由于 SLBT 使得路径较长，因此平均延时较大。经计算，相比 DLBT 和 SLBT，LDG-ACO 算法的平均延时分别降低了约 75%和约 52%，说明本书提出的 LDG-ACO 算法的负载均衡效果更好，使得网络延时有明显下降。

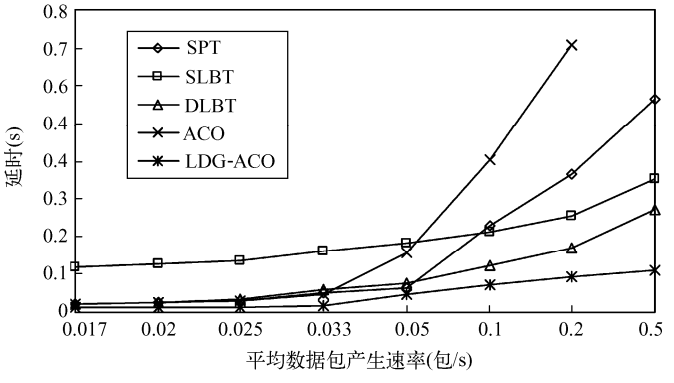


图 4.7　网络延时比较

Fig. 4.7　Compare of packet delay

4．负载均衡因子

负载均衡因子随时间变化情况如图 4.8 所示，标准 ACO 路径收敛的特点使得其负载均衡因子为 0.18，并不随时间变化，相比其他算法最小。SPT 算法和 SLBT 算法的负载均衡程度不随时间变化而变化，其中 SPT 算法负载均衡因子

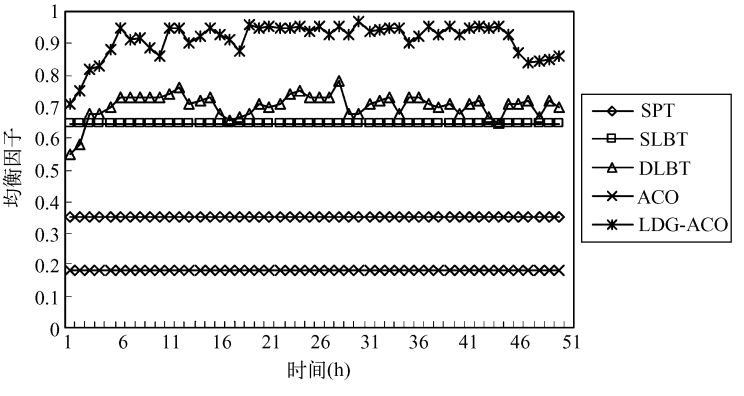


图 4.8　负载均衡因子比较

Fig. 4.8　Compare of load balancing factor

为 0.35，SLBT 算法由于没有考虑各个节点数据采集量的差异，因而负载均衡程度不如 DLBT 算法和本书提出的 LDG-ACO 算法。DLBT 算法的负载均衡因子的平均值为 0.7，本书提出的 LDG-ACO 算法的负载均衡因子的平均值为 0.85，相比 DLBT 提高了约 21%。由此可见，本书提出的 LDG-ACO 算法的负载均衡程度较其他四种算法更好。

5．网络寿命

本次仿真将网络寿命定义为从网络开始运行到出现第一个节点“死亡”的时间。图 4.9 显示了在不同的数据发送速率下的网络寿命，本书提出的 LDG-ACO 算法的寿命相对于其他四种算法更长。经计算，相比 ACO、SPT、SLBT 和 DLBT 算法，LDG-ACD 算法分别提高了网络寿命约 200%、约 120%、约 81%和约 43%。

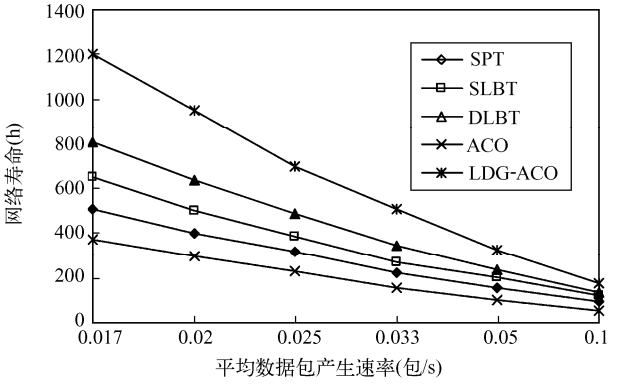


图 4.9　网络寿命比较

Fig.4.9　Compare of network lifetime

综上所述，从以上各项性能指标结果表明，本章提出的 LDG-ACO 算法比其他四种典型方法效果更优。

4.5　小结

在无线传感器网络数据汇集应用中，当数据源产生数据的速率随时间而变化，通过构造负载均衡数据汇集树的方法不能够达到负载均衡的效果时，需要采用动态负载均衡的方法。本章基于蚁群优化的原理，将无线传感器网络的数据汇集抽象为蚂蚁搬运数据的动态过程，并改进了标准的蚁群优化，使之适用于要求负载均衡的数据汇集应用。仿真实验表明，相比相关算法，本书提出的 LDG-ACO 算法在网络性能和网络寿命方面有显著的提高。另外，LDG-ACO 算

法无须位置获取、功率控制等其他手段支持，对网络节点和应用环境要求低，使得其更加实用。LDG-ACO 算法不仅可以应用于无线传感器网络，还可以应用于其他类似的数据汇集网络。

参 考 文 献

[1] Dorigo M, Stutzle T. Ant colony optimization[M]. MA : MIT Press, 2004.

[2] Chandrasekar R, Misra S. Introducing an ACO Based Paradigm for Detecting Wildfires using Wireless Sensor Networks[C]. Ad Hoc and Ubiquitous Computing, 2006. ISAUHC '06. International Symposium on. 2006: 112-117.

[3] Yixiong C. Load Balancing in Non-dedicated Grids Using Ant Colony Optimization[C]. Semantics, Knowledge and Grid, 2008. SKG '08. Fourth International Conference on. 2008: 279-286.

[4] 廖新飞, 陶利民. 基于多态蚁群系统的无线传感器网络数据聚集算法[J]. 计算机应用. 2007(08): 1849-1851.

[5] 耶刚强, 梁彦, 孙世宇, et al. 基于蚁群的无线传感器网络路由算法[J]. 计算机应用研究. 2008(03): 715-717.

[6] 高利, 李仁发, 罗娟. 基于蚁群算法的传感器网络分布式广播算法[J]. 计算机工程. 2007(13): 135-137.

[7] 王结太, 许家栋, 徐建城. 基于蚁群优化算法的无线传感器网络路由协议[J]. 系统仿真学报. 2008(18): 4898-4901.

[8] 张曦煌, 夏佳, 沈玉方. 一种蚁群竞争WSN能量均衡路由算法[J]. 计算机应用. 2007(08): 1825-1827.

[9] Yang H, Ye F, Sikdar B. A dynamic query-tree energy balancing protocol for sensor networks[C]. Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE. 2004: 1715-1720.

第 5 章　移动执行器动态负载均衡数据汇集算法

随着智能手机、掌上电脑等手持设备的发展，越来越多的应用需要将无线传感器网络的执行器节点与手持设备整合，要求无线传感器网络的数据汇集算法支持执行器节点移动，这是对无线传感器网络数据汇集算法设计的一个挑战。

本章针对无线传感器网络移动执行器数据汇集应用，研究负载均衡数据汇集机制，提出一种无线传感器网络移动执行器负载均衡数据汇集算法（Load-balanced Data Gathering algorithm for Mobile 执行器，LDG-MS）。LDG-MS 算法借鉴群体智能[1]思想，将无线传感器网络节点视为仅有简单智慧的个体，通过定义两个简单的规则，对无线传感器网络节点的数据转发行为进行描述，最终把数据汇集行为解释为所有个体（节点）在简单规则下的智慧涌现。LDG-MS 算法将下跳节点的决策问题抽象成一个多目标规划问题，并采用距离加权法求解。同时，LDG-MS 算法采用功率控制策略解决由于执行器节点移动造成的链路中断问题，并给出了详细的计算方法。仿真实验表明，LDG-MS 算法在网络性能和网络寿命方面较相关方法有显著提高。

5.1　执行器节点移动对网络数据流模型的影响

根据无线传感器网络不同的应用环境，可以将无线传感器网络的数据流传输模型分为三种基本类型：连续型数据流模型、查询型数据流模型和事件型数据流模型。

5.1.1　连续型数据流模型

在绝大多数环境监测（如流域水质监测、大气监测、城市噪声监测等）、交通流量监控、工业现场设备监控、油气水管道流量监控、农业大棚温室度监测、水电气远程抄表和远程医疗等应用场景中，通常需要传感器节点周期性地向监控室（或用户）汇报监测数据。各个传感器节点的数据源源不断地向汇聚节点（执行器节点）传输，本书称这种数据流模型为连续型数据流模型，如图 5.1 所

示。连续型数据流模型应用即为数据汇集应用。

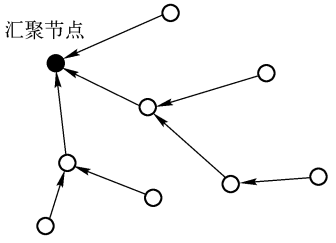


图 5.1　连续型数据流模型

Fig. 5.1　Continues data stream model

连续型数据流模型的特点是数据流量大、数据流连续、多点向一点汇集（多对一）。连续型数据流通常采用树状路由拓扑结构。首先由执行器节点发起一个路由建立消息，然后该消息被网络节点不断转发，形成一棵路由树，最后大量、连续、稳定的数据流沿着路由树反方向向执行器节点汇集。由于连续型数据流模型的数据流连续，一旦执行器节点移动，将很可能破坏路由树的结构，导致数据在传输给执行器节点的最后一跳丢失。因此，一般情况下，连续型数据流模型不能够很好地支持执行器节点移动。

5.1.2　查询型数据流模型

在矿井人员位置监测、车辆位置监测和库房产品存储情况监测等应用场景中，通常用户不需要实时了解各个监测节点的状况，而只在用户需要得到某个（或某些）节点的数据时，才主动通过服务接入的执行器节点向网络注入查询消息，被查询的节点收到该消息后，反馈查询结果。由于查询操作是由用户临时发起的，网络通常在查询时建立临时路由。在无其他辅助信息（如位置信息）的情况下，查询消息通常将被传遍全网，形成一棵查询路由树，如图 5.2 中虚线所示。查询得到的反馈数据沿着路由树的反向路径传回执行器节点，如图 5.2 中实线所示。通常称由用户发起的查询数据流模型为查询驱动型数据流模型，简称查询型数据流模型。

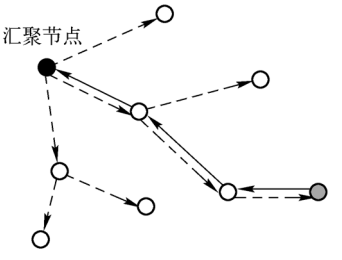


图 5.2　查询驱动型数据模型

Fig. 5.2　Query-driving data stream model

查询驱动型数据流模型的特点是数据量小、用户驱动、路由临时建立。由于查询路由是临时建立的，传输完成后撤销，因此，查询驱动型数据流模型能够较好地支持执行器节点移动。

5.1.3　事件型数据流模型

在火灾监测、地震监测和社区安防等应用中，通常只需网络节点在监测到突发事件（如发生火灾、地震或非法入侵等）才向用户报警。由于突发事件发生的

频度较低，并且事件发生的时间和地点是不可预知的，因此通常不需要事先建立和维护路由。当突发事件发生后，监测到事件的节点通过一定方式（如网格传输）将事件数据传输给执行器节点，从而使用户收到事件报警信息。通常称这种事件数据流模型为事件驱动型数据流模型，简称事件型数据流模型，如图 5.3 所示。

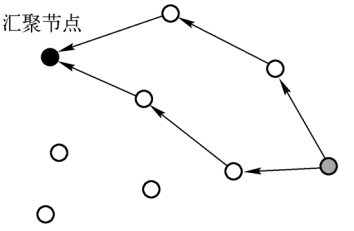


图 5.3　事件驱动型数据流模型

Fig. 5.3　Event-driving data stream model

事件驱动型数据流模型的特点是数据量小、突发性强、路由临时建立。由于采用临时建立路由的方式，因此，事件驱动型数据流模型能够较好地支持执行器节点移动。

综上所述，查询型数据流模型和事件型数据流模型的路由是临时性的，执行器节点的移动对数据传输的影响不大。而在连续型数据流模型中，通常由执行器节点事先建立路由，如 TEEN、APTEEN、MINA[2]，而执行器节点的移动会造成事先建立的数据传输链路中断，如果频繁地重新建立路由，不仅网络能耗代价比较大，而且大量的广播消息还容易造成网络风暴，阻碍正常的数据传输。连续型数据流模型应用即为数据汇集应用，因此，执行器节点的移动对数据汇集应用的影响很大，是设计无线传感器网络数据汇集算法的一个挑战。

5.2　LDG-MS 算法思路

当执行器节点在移动过程中，其他节点向执行器节点发送数据前，必须知道执行器节点当前所处方位，以及自身的哪个邻居节点距离执行器节点最近，才能正确地选择数据的下跳节点。因此，移动执行器数据汇集问题就抽象为移动目标发现问题。群体智能在解决移动目标发现问题上具有独特优势。群体觅食行为作为群体智能的一种典型的表现形式，其规则相当简单。在群体觅食过程中，只要某个或某些个体发现食物，其他个体就会通过察觉邻域个体的行为，跟随邻域个体找到食物，从而使得整个群体找到食物。无线传感器网络单个节点的计算能力、存储器空间、能量和通信半径都有限，使得每个节点只能感知邻居节点，并与之发生信息交互，这与群体智能中的个体情况非常相似。因此，用群体智能的思想来解决无线传感器网络中的相关问题是非常契合的。在无线传感器网络移动执行器数据汇集应用中，可以将执行器节点视为移动的食物源，将数据汇集过程看作群体觅食行为。

基于 LDG-MS 算法的数据汇集过程可以描述为：执行器节点在移动过程中周期性地发送信标，告知周围的邻居节点自身目前所处的位置。执行器节点的邻居节点（即内层节点，如图 5.4 所示）能够直接发现执行器节点，并开始传输数据，处于外层的节点通过感知内层节点的数据传输行为，间接发现执行器节点，并传输数据。外层节点能够发现移动执行器节点位置的关键在于节点在数据传输过程中附带报告当前执行器节点的位置信息，其外层节点“偷听”邻居节点的数据传输信息，间接获知执行器节点的位置，以此层层递进，从而使得网内所有的节点发现执行器节点，并跟踪执行器节点的位置变化。为了使网络节点能量均衡消耗，外层节点在数据传输过程中，通过估计邻居节点距执行器节点的距离和邻居节点的负载情况，进行下一跳的路由选择。

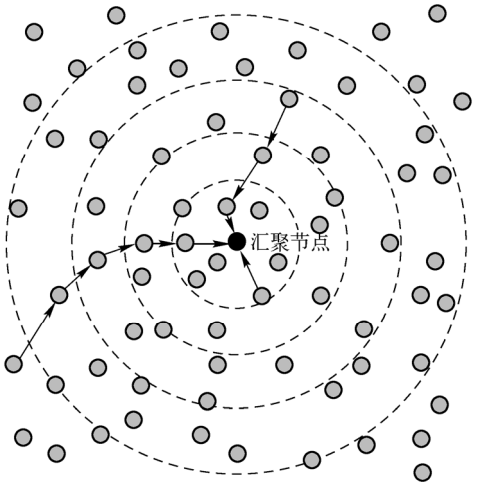


图 5.4　节点的层次

Fig. 5.4　The level of nodes

5.3　LDG-MS 算法描述

5.3.1　LDG-MS 算法规则与定义

本章将无线传感器网络的非执行器节点（即普通传感器节点）称为 Node 节点。

LDG-MS 算法定义两种消息：

（1）sink_BEACON(*sinkInfo*)：表示执行器节点通告邻居节点，其中 *sinkInfo*

包括执行器节点所处的位置（x, y）和一个表示位置信息新旧程度的序列号 *seq*，*seq* 为递增顺序编号。

（2）SENSOR_DATA(*nextAddr*, *data*, *sinkInfo*，*loadInfo*)：表示将数据内容（*data*），通过下跳节点（*nextAddr*）转发给执行器节点，并附带自身最新收到的执行器节点的位置信息（*sinkInfo*）和节点自身的负载信息（*loadInfo*）。

Node 节点保存执行器节点的位置信息，当 Node 节点收到一个新的执行器节点位置信息（即含有新序列号的位置信息）时，立即对其更新。Node 节点根据收到 sink_BEACON 消息的情况，改变自身状态。对于 Node 节点的状态有如下定义：

【定义 5.1】对于 Node 节点，如果连续超过时间 T（sink_BEACON 消息周期）没有收到执行器节点发送的信标消息（sink_BEACON），则称该 Node 节点的状态为非执行器邻居节点状态，记为 NSA（Non-Sink Adjacent）。否则称该 Node 节点的状态为执行器邻居节点状态，记为 SA（Sink Adjacent）。

Node 节点在不同的状态时，具有不同的数据转发规则：

【规则 5.1】如果 Node 节点为 SA 节点，则直接将数据传输给执行器节点。

【规则 5.2】对于任意一个 NSA 节点，选择下一跳 Node 节点的过程中存在两个准则：（1）尽可能选择距离执行器节点近的 Node 节点；（2）尽可能选择负载低的 Node 节点。

在理想情况下，距离执行器节点越近的 Node 节点具有更少的传输跳数，从而有更少的传输延迟和能量消耗。而选择负载低的 Node 节点作为下跳节点是为了均衡网络负载的消耗，延长网络寿命。

任意节点 i 与执行器节点间的距离计算：

$$d_i = \sqrt{(x_i - x_{\text{sink}})^2 + (y_i - y_{\text{sink}})^2} \tag{5.1}$$

对于任意一个 NSA 节点 i，尽可能选择距离执行器节点近的节点 j 作为下跳节点，可以表示为

$$d(j) = \min_{j \in N(i)} \left(\sqrt{(x_j - x_{\text{sink}})^2 + (y_i - y_{\text{sink}})^2} \right) \tag{5.2}$$

式中，$N(i)$表示节点 i 的邻居节点。

无线传感器网络节点的负载量化采用第 5 章的方式：

$$l_i = \frac{1}{\lambda_e \dfrac{e_i}{E_i} + \lambda_q \dfrac{q_i}{Q_i}} \tag{5.3}$$

式中，l_i 为节点 i 的负载；q_i 为节点 i 的空闲队列长度；Q_i 为为节点总队列长度；e_i 为节点 i 的剩余能量；E_i 为节点 i 的总能量或初始能量；λ_e 和 λ_q 分别为能量和

队列的权重。消息在队列中会产生排队延时，因此，能量和队列的权重由实际应用对实时性的要求确定。

对于一个 NSA 节点 i，尽可能选择负载低的节点 j 作为下跳节点，可以表示为

$$l(j)=\min_{j\in N(i)}\left(\frac{1}{\lambda_e\frac{e_i}{E_i}+\lambda_q\frac{q_i}{Q_i}}\right) \tag{5.4}$$

因此，对于一个 NSA 节点 i，如何选择下一跳节点 j 的问题转化为一个有约束的多目标规划问题：

$$\begin{aligned} &F(j)=\min_{j\in N(i)}(d(j),l(j))^{\mathrm{T}}\\ &\text{s.t.}\qquad d_j<d_i \end{aligned} \tag{5.5}$$

式中，约束条件 $d_j<d_i$ 避免了数据转发回环。

处理多目标规划问题时，通常采用构造评价函数的方法将多个目标问题转化为一个单目标问题来求解。本章采用距离加权求和法[3]，其基本原理为：对于一个标准的多目标问题 $\min F(X), X\in R, R=\{X\mid g_i(X)\geqslant 0, h_j(X)=0, i=1,2,\ldots,m; j=1,2,\ldots,n\}$，给出一组数 f_1^0，$f_2^0,\ldots,f_p^0$，分别为单个目标规划问题 $\min_{x\in R} f_i(X)$，$i=1,2,\ldots,p$ 的下界，即满足 $\min_{X\in R} f_i(X)\geqslant f_i^0$ $(i=1,2,\ldots,p)$，采用的评价函数为 $\min_{X\in R} h(F(X))=\sum_{i=1}^{p}|\lambda_i f_i(X)-f_i^0|$，权系数 λ_i 的值由各目标函数 $f_i(X)$ 的重要程度给出。

【定义 5.2】对于规则 5.2 的两个准则，定义其权系数分别为 λ_d 和 λ_l，$\lambda_d>0$，$\lambda_l\geqslant 0$，$\lambda_d+\lambda_l=1$。λ_d 和 λ_l 分别代表了与执行器节点距离的重要程度和节点负载均衡的重要程度。

求解式（5.5）的评价函数为

$$\begin{aligned} &h(F(j))=\min_{j\in N(i)}(\lambda_d\mid d(j)-d^0\mid+\lambda_l\mid l(j)-l^0\mid)\\ &\text{s.t.}\quad d_j<d_i,\lambda_d>0,\lambda_l\geqslant 0,\lambda_d+\lambda_l=1 \end{aligned} \tag{5.6}$$

式中，d^0 为最短距离；l^0 为最小负载；权系数 λ_d 和 λ_l 的确定与具体的应用环境相关。在执行器节点移动快时，或者应用要求实时性较高时，增加 λ_d 的权重；在执行器节点移动慢时，或者应用要求能量均衡性较高时，增加 λ_l 的权重。

5.3.2 功率控制策略

执行器节点在移动过程中以一定的周期发送 sink_BEACON 消息，Node 节

点根据是否收到 sink_BEACON 消息而改变自身状态，并确定数据转发规则。但是，不论执行器节点以多高的速率发送 sink_BEACON 消息，总会在一定的时段内（即两次 sink_BEACON 消息之间），可能由于自身移动造成与 SA 节点的链路中断，导致在这段时间内出现丢包。如图 5.5 所示，假设网络中节点的通信半径相同，节点 *J* 和执行器节点都处于对方的通信半径的边缘（实线为节点 *J* 发送 SENSOR_DATA 消息的范围，虚线为执行器节点发送 sink_BEACON 消息的范围）。此时节点 *J* 仍可以收到 sink_BEACON 消息，因此其状态仍为 SA 节点，但只要执行器节点继续沿着图示方向运动，就会造成 *J*→执行器的链路中断。在这种情况下，至少需要等待一个 sink_BEACON 消息周期，才能使节点 *J* 转变到正确状态。由此可见，在相同通信半径的情况下，移动的执行器节点在与其通信范围边缘的节点建立了路由关系，容易造成链路中断。

为了解决上述问题，本章提出对不同类型的消息进行功率控制，使 sink_BEACON 消息半径小于 SENSOR_DATA 消息半径。如图 5.6 所示，由于执行器节点发送 sink_BEACON 消息的半径小于节点 *J* 发送 SENSOR_DATA 消息的半径，即使执行器节点与其 sink_BEACON 消息半径边缘的节点 *J* 建立了邻居关系，执行器节点仍然可以继续沿着图示方向移动一定的距离而不会导致 *J*→执行器的数据传输链路中断。只要执行器节点未移动出节点 *J* 的通信半径，节点 *J* 的 sink_BEACON 消息的有效期超时就可以使其状态得到正确的转变，从而避免由于执行器节点移动导致链路中断所产生的丢包和延时。目前，大部分无线传感器网络节点的射频收发控制器都支持功率控制功能，如 CC2430、CC1100 和 MC132x 等。

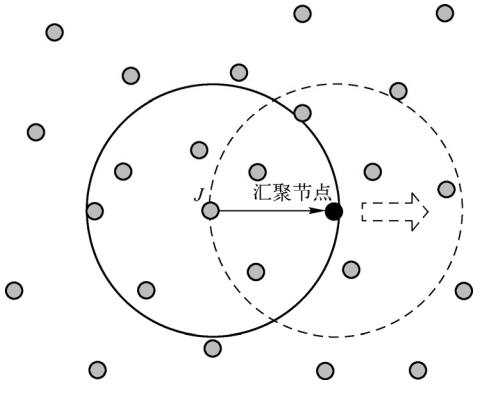


图 5.5　信号边缘节点

Fig. 5.5　Node at signal edge

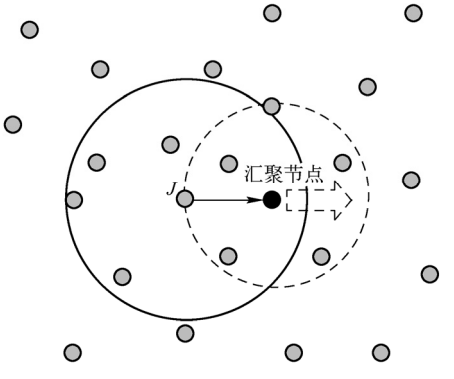


图 5.6　sink_BEACON 消息功率控制

Fig. 5.6　Power control of sink_BEACON

由此可见，sink_BEACON 消息半径越小，执行器节点发送 sink_BEACON 消息的周期就可以越长，执行器节点就越省电。但过小的 sink_BEACON 消息通信半径会造成网络不连通，无法建立 SA 节点，如图 5.7 所示。因此，应该满足执行器节点的 sink_BEACON 消息覆盖范围内有节点分布，才能使网络存在 SA 节点。

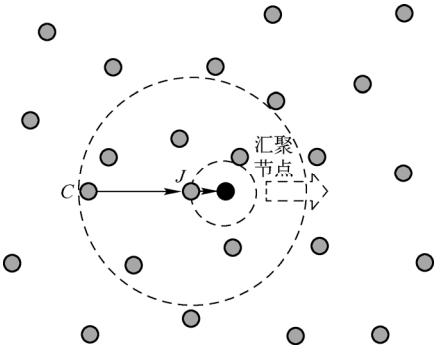


图 5.7　过小的 sink_BEACON 消息半径

Fig. 5.7　Too small communication radius of sink_BEACON

怎样确定执行器节点发送 sink_BEACON 消息的半径才合理呢？假设网络节点均匀分布，密度为 ρ，执行器节点发送 sink_BEACON 消息的半径为 r。文献[4]证明了均匀分布的无线传感器网络在任意区域内的节点数目服从泊松分布，即在面积为 S 的任意区域中，包含 X 个节点的概率为

$$P\{X=m\}=\frac{(\rho\cdot S)^m\cdot \mathrm{e}^{-\rho\cdot S}}{m!} \tag{5.7}$$

根据式（5.7），求解半径 r 的问题转化为

$$P\{X>0\}=1-P\{X=0\}\geqslant\alpha \tag{5.8}$$

式中，α为可靠程度，表示出现 SA 节点的可能程度。例如，α取值为 0.99，表示保证 99%的几率出现 1 个及其以上的 SA 节点。

由式（5.8）可以推出

$$1-\frac{(\rho\cdot\pi r^2)^0\cdot \mathrm{e}^{-\rho\cdot\pi r^2}}{0!}=1-\mathrm{e}^{-\rho\pi r^2}\geqslant\alpha,(r>0) \tag{5.9}$$

由式（5.9）可得半径 r 计算为

$$r\geqslant\sqrt{\frac{\ln(1-\alpha)}{\pi\rho}} \tag{5.10}$$

5.3.3　Sink_BEACON 消息周期计算

按照 5.3.2 节所述，应当在执行器（sink）节点还未移动出邻居节点通信半

径范围时使 sink_BEACON 消息的有效期超时。假设执行器节点的平均移动速度为 V，Node 节点发送 SENSOR_DATA 消息的半径为 R，执行器节点发送 sink_BEACON 消息的半径为 $r(r<R)$。如 5.8 所示，sink_BEACON 消息周期 T 应该满足：$VT \leqslant R-r$。

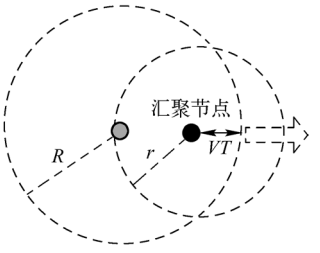


图 5.8　sink_BEACON 消息周期

Fig. 5.8　Period of sink_BEACON

因此，sink_BEACON 消息周期 T 计算如下：

$$T \leqslant \frac{R-r}{V},(r<R) \qquad (5.11)$$

例如，Node 节点发送 SENSOR_DATA 消息的半径为 100m，执行器节点发送 sink_BEACON 消息的半径为 50m，执行器节点的平均移动速度为 10m/s（人类的最大步行速度），由式（5.11）可得 sink_BEACON 消息的最大周期为 5s。

5.3.4　LDG-MS 算法伪代码

执行器节点主要负责发送信标和数据汇集任务，其伪代码如下：

```
/*Pseudo-code of 执行器*/
timerStart(T, PERIOD_TYPE);              //开启周期性定时器，定时周期为 T
while (haveEnergy){
  if (timerFired){
    sendMsg( sink_BEACON, sinkInfo);     //发送位置信息
  else if (receivedMsg)；
    renderMsg();                         //向应用层提交数据
 }
}
```

Node 节点主要负责数据采集和转发任务，且 LDG-MS 算法为分布式算法，Node 节点的伪代码如下：

```
/*seudo-code of Node*/
Status = NSA;                            //初始化为 NSA 节点
while (haveEnergy){
  switch (receivedMsg){
    case sink_BEACON:
      status = SA;                       //成为 SA 节点
      recordsinkInfo();                  //记录执行器节点信息
      timerStart(T, SINGLE_TYPE);        //开启单次定时器，定时周期为 T
```

```
        break;
        case SENSOR_DATA:
          if (toSelf){                          //收到上跳节点数据
            computeNextHop();                   //根据转发规则计算下跳节点
            forward();                          //转发数据
          }
          else
            recordInfo();                       //记录执行器节点和邻居节点信息
        break;
        default: break;
    }
    Pseudo-code of Node
      }

    /*定时中断*/
    if (timerFired) {                           //超时
      status = NSA;                             //成为 SA 节点
    }

    /*传感器中断*/
    if (sensorDataReady){
      computeNextHop();                         //根据转发规则计算下跳节点
      sendData();                               //发送数据
    }
```

5.4 仿真验证

5.4.1 仿真环境与参数

仿真实验使用 NS-2 平台。网络节点数设置为 401，其中 1 个为执行器节点，其他 Node 节点均匀分布在 800m×800m 的平面区域。执行器节点随机在网络内部移动，其移动速度为 10m/s，Node 节点的初始能量设置为 50J，执行器节点的能量无限制，$\lambda_e=\lambda_q=1$，其他仿真参数由表 5.1 列出。为了简化计算，定义节点用于感知、接收与传送一个字节数据所耗费的能量分别为 1×10^{-5}J、5×10^{-5}J 和 1×10^{-4}J。由式（5.10）计算得到执行器节点发送 sink_BEACON 消息的半径 $r\geqslant50$m，r 取值为 80m。设置 Node 发送 SENSOR_DATA 消息的半径为 150m，由式（5.11）计算得到 sink_BEACON 消息的周期 $T\leqslant7$s，T 取值为 5s。

表 5.1　仿真参数

Table 5.1　Simulation parameters

参　　数	值
Scene size	800m×800m
Node number	400 node + 1 sink
Mac	802.11
Application	CBR
Packet Size	1024
Queue length	10
Parameter	Value
Channel model	TwoRayGround

5.4.2　功率控制

为了控制执行器节点发送 sink_BEACON 消息的功率和 Node 节点数据传输的功率，根据 TwoRayGround 模型[5]计算发送功率如下：

$$P_t = \frac{P_r \cdot d^4 \cdot L}{G_t \cdot G_r \cdot h_t^2 \cdot h_r^2} \tag{5.12}$$

式中，P_r 为接收功率门限；d 为传输距离；L 为天线长度；h_t 为发送天线高度；h_r 为接收天线高度；G_t 为发送天线增益；G_r 为接收天线增益。根据 Phy/WirelessPhy 网络物理接口默认的接收门限 RxThreash_为 3.65262e-10，以及天线长度、高度和发送接收增益，由式（5.12）计算得到仿真所使用的传输距离与发送功率的对应关系，见表 5.2。

表 5.2　传输距离与发送功率

Table 5.2　Transition distance VS TxPower value

距离（m）	发送功率（W）
80	3.42653e-3
150	3.65362e-2

为了对不同的分组进行功率控制，在 wireless-phy.cc 的 sendDown 函数中需对分组的类型和本地地址进行识别，并控制节点的发送功率变量 P_t。

5.4.3　仿真结果

本次仿真实验主要对比了与本章应用环境相似的算法，即文献[111]和文献[112]所提出的方法，分别命名为 sink_Claim 和 SLM。

1．网络吞吐量（throughput）

在网络吞吐量性能方面，如图 5.9 所示，本书提出的 LDG-MS 算法的吞吐量最大，其次是 sink_Claim，SLM 的吞吐量最小，其中横坐标表示包平均

产生间隔（average packet generate interval）。经计算，相比 sink_Claim 和 SLM，LDG-MS 的吞吐量分别提高了约 13%和约 57%，说明本书提出的 LDG-MS 的负载均衡效果较好，使得网络吞吐量有明显提高。

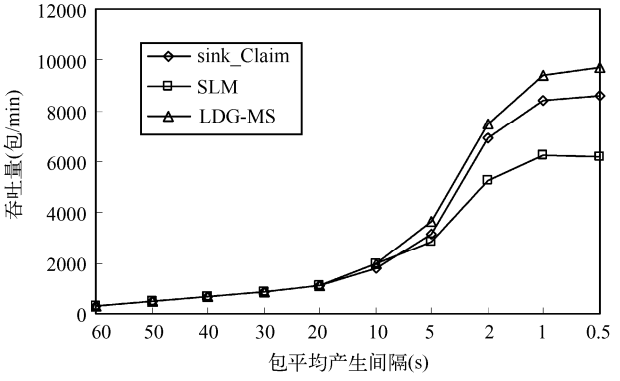


图 5.9　吞吐量比较

Fig. 5.9　Compare of network throughput

2．丢包率（packet loss rate）

在网络丢包率方面，如图 5.10 所示，本书提出的 LDG-MS 的丢包率最小。随着发送速率的增大，各个算法的丢包率都有所增加，但 LDG-MS 的丢包率增幅明显小于 sink_Claim 和 SLM。经计算，相比 sink_Claim 和 SLM，LDG-MS 的平均丢包率分别降低了约 44%和约 68%，说明本书提出的 LDG-MS 的负载均衡效果较好，使得丢包率有明显下降。

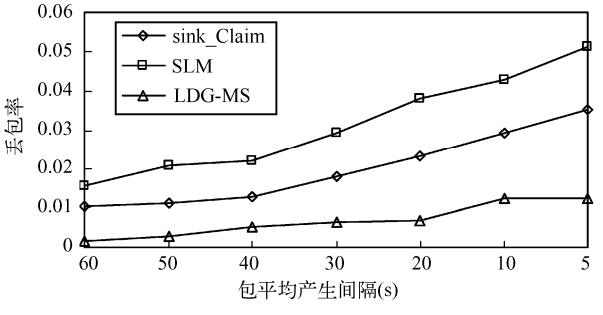


图 5.10　丢包率比较

Fig. 5.10　Compare of packet loss rate

3．网络延时

在仿真过程中，根据时间顺序抽取了一个边缘节点的数个 CBR 流，观察其传输延迟。如图 5.11 所示，SLM 方法中的每次路由查询操作，使得网络延时

维持在一个相对较高的水平。sink_Claim 方法中，由于执行器节点移动，路由路径不断增加，网络延时也不断增大。本章提出的 LDG-MS 算法相对前两种方法延时最小，且延时不会因执行器节点的移动而增加。经计算，相比 sink_Claim 和 SLM，LDG-MS 的网络平均分别降低了约 68%和约 74%，说明本书提出的 LDG-MS 算法使得网络延时有明显下降。

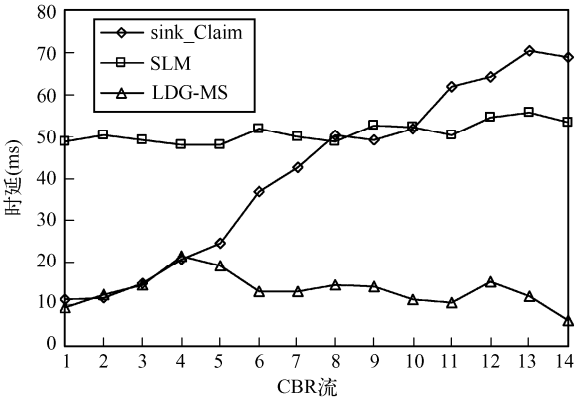


图 5.11　时延比较

Fig. 5.11　Compare of packet delay

4．网络寿命

本次仿真将网络寿命定义为从网络开始运行到出现第一个节点“死亡”的时间。网络寿命情况，如图 5.12 所示。SLM 方法中，每个节点在发送数据包时，都要通过与 SLM 节点进行交互查找执行器节点，这使得 SLM 节点及其邻居节点的能量快速消耗，严重影响网络寿命，因此其网络寿命最短。sink_Claim 方法在负载均衡上没有做任何优化，因此其网络寿命不及 LDG-MS

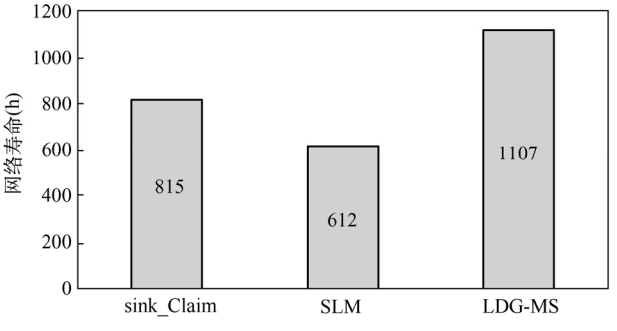


图 5.12　网络寿命比较

Fig. 5.12　Compare of network lifetime

算法。经计算，相比 sink_Claim 和 SLM，本书提出的 LDG-MS 的网络平均分别增加了了约 36%和约 81%，说明本书提出的 LDG-MS 算法使得网络寿命有明显提高。

5.5 小结

本章针对无线传感器网络数据汇集应用，研究执行器节点移动情况下的数据汇集机制，在群体智能思想的启发下，提出了 LDG-MS 算法。LDG-MS 算法通过定义两个简单的规则，对无线传感器网络节点的数据转发行为进行描述，将下跳节点的决策问题抽象成一个多目标规划问题，并对数据传输的实时性和节点负载进行加权求解，以满足不同的应用中对实时性和能耗均衡的需求。为了解决由于执行器节点移动造成的链路中断问题，本章提出对不同的消息类型进行功率控制，并给出了详细的计算方法。仿真实验表明，相比相关算法，本书提出的 LDG-MS 算法在网络性能和网络寿命方面有显著的提高。

本章在求解最优下一跳的过程中，仅考虑了 1 跳范围内的优化，属于一种贪心策略，而对于多跳传输的无线传感器网络系统的全局优化是一个非常复杂的问题，有待进一步研究。

参 考 文 献

[1] 张铃, 程军盛. 松散的脑袋——群体智能的数学模型[J]. 模式识别与人工智能. 2003(1): 1-5.

[2] Ding J, Sivalingam K, Kashyapa R, et al. A multi-layered architecture and protocols for large-scale wireless sensor networks[C]. Vehicular Technology Conference, 2003. VTC 2003-Fall. 2003 IEEE 58th. 2003.

[3] 范玉妹, 许尔, 周汉良. 数学规划及其应用[M]. 冶金工业出版社, 2004.

[4] 崔逊学, 方红雨, 朱徐来. 传感器网络定位问题的概率特征[J]. 计算机研究与发展. 2007(04): 630-635.

[5] Rappaport T S. Wireless Communications: Principles and Practice[M]. IEEE Press Piscataway, NJ, USA, 1996.

第6章　基于SA协作的分簇算法

在无线传感器/执行器网络中，为了减少网络通信负载，同时避免过多的执行器节点同时参与同一事件的处理，通常需要将网络节点划分为若干簇（cluster）。每个簇以功能强大的执行器节点为簇头，其他传感器节点为簇成员。当传感器节点监测到事件发生后，只向本簇簇头报告，由簇头决定采取相应行动，由此形成传感器节点与执行器节点之间基于分簇的协作模型，即簇内 SA 协作模型。模型必须考虑构成网络的执行器节点理想数量，传感器节点之间以及与执行器节点应该采取的通信半径，评估事件发生频率、执行任务能耗等因素对此的影响。

6.1　SA协作模型特点

与无线传感器网络的大多数以同构节点（假定网络内所有节点均相同）为目标的分簇算法不同，由于执行器节点的加入，使得算法具有以下不同特点：

1．支持异构性

在无线传感器/执行器网络中，传感器节点与执行器节点的作用和能力不同，导致网络可能划分为不同层次。如果簇头由执行器节点充当，则可分为传感器节点与执行器节点、执行器节点与执行器节点两个层次。如果簇头仍然在传感器节点中产生，由簇头控制执行器节点，则会变成传感器节点与簇头、簇头与执行器节点、执行器节点之间三个层次。

2．成簇过程

在无线传感器网络中，由于节点能量有限，往往轮流充当簇头，一旦充当簇头的节点能量低于某个阈值，就需要重新进行分簇，在整个网络生存期中，可能经历多次分簇过程，簇头与簇成员每次都要发生变化。而无线传感器/执行器网络，由于执行器节点功能更强，能量也更充足，同时负有执行任务的特殊使命，可以固定充当簇头，也可在传感器节点中选出簇头，由簇头对执行器节点发出命令。无线传感器网络由各节点按照概率或者剩余能量等条件轮流充当簇头，分簇过程在每次轮换时都要进行，影响正常的数据采集。而无线传感器/执行器网络中如果由执行器节点充当簇头，只需部署阶段由执行器节点发出申明，传感器节点加入即可，整个生存期只有一次成簇过程。

3．事件发生频率和任务对簇的影响

无线传感器网络节点的任务主要是监测环境数据，分簇算法只需考虑收发传感数据能耗。而无线传感器/执行器网络由于执行器节点可能根据事件类型采取行动，必须考虑事件发生频率和执行任务能耗对分簇的影响。

4．多个应用目标对分簇的影响

由于无线传感器/执行器网络分簇结构固定，传统分簇算法中强调的连通度、覆盖度因素变得不再重要，而实时性、能量有效性等应用要求成为分簇算法优化的重点。

显然，无线传感器/执行器网络的分簇部署算法，必须基于以上特点进行设计。

6.2 无线传感器/执行器网络分簇算法分析

文献[1]研究了分簇算法和功率控制策略对最小化网络延迟的影响，提出一种能量感知的实时分簇路由协议（RECRP），该协议由“网络一次成簇”、“网络二次成簇与邻居节点管理”、“路由生成”、“传感器簇首节点轮换”等四部分组成。RECRP 协议首先使无线传感器/执行器网络中的传感器节点完成一次成簇，接着，推选出的传感器簇首节点与执行器节点完成网络的二次成簇，使整个网络形成一个由“传感器从节点-传感器簇首”、“传感器簇首-执行器节点”和“执行器节点-执行器节点”组成的三层网络架构。网络完成分簇工作后，协议采用实时相关、能量有效策略进行消息的传递，当一轮数据传输结束后，协议通过簇首节点轮换机制使各传感器节点均匀消耗能量。然而，该协议在第三阶段，采取传感器节点使用最小功率发送数据，而执行器节点以最大功率发射，这一策略显然会在多跳簇内造成转发时延，发射功率的确定并没有从能耗与时延角度进行严格计算；同时，每轮数据传输结束，就要进行簇头轮换，造成额外开销。Ghalib A. Shah 等人提出的实时协同与路由框架（RCR）[2]，由分簇协议动态加权分簇算法（DAWC）和延迟约束的能量感知路由机制（DEAR）两部分构成。与 RECRP 协议不同，DAWC 既没有周期性分簇过程，也没有固定跳数的簇规模，而是根据动态变化的网络拓扑自适应。DAWC 的第一阶段是构建理想节点数的簇，在此期间，簇头得到每个簇成员的时延，用以选择发送数据的恰当节点。簇的选举过程根据权值公式，计算各节点的权值，并选出最大权值的节点充当簇头。但是在计算簇理想个数时，协议只考虑了分簇过程产生的能耗，并没有从全网的传感数据报告和执行器节点执行相应任务能耗的角度建立能耗公式，并由此计算簇理想节点数。Neeta Trivedi 等人提出的 Ripples

分簇协议[3]，是最小化消息复杂度的载荷均衡的分簇协议，该协议主要关注传感器节点的自适应覆盖。此外，执行器不再充当簇头角色而由传感器轮流充当，以提高网络生存期。协议的新颖之处在于利用最低可能的控制开销保持网络理想的覆盖度，同时，优化多余部署，产生平衡的簇。然而，在无线传感器/执行器网络中，正如本书关于 SA 协作模型特点提及的那样，覆盖度并不像无线传感器网络分簇算法那么重要，其他如实时性和能耗均衡等应用要求，应该在分簇算法中重点考虑，这正是该协议薄弱之处。文献[4]提出一种基于传感器部署与连通度的分簇算法，以执行器簇头，并按照执行器覆盖最大和数据汇集时间最小原则进行部署，保证传感器网络的 k-hop 独立控制集（IDS）。算法分为两个阶段：传感器首先根据 IDS 选出簇头，然后执行器再部署到选出的簇头位置。并利用分簇算法保证执行器之间的连通度。在文献[5]中，执行器统一部署在监控区域，然后在每个簇内基于传感器的顶点利用 1-center 原理重新定位，每个传感器以最近的执行器为簇头。上述两种算法关注的重点仍然是连通度，没有从无线传感器/执行器网络的多种应用要求的角度考虑分簇结构中传感器节点的通信方式和执行器部署问题。

本章针对上述算法的不足之处，提出一种基于 SA 协作的分簇算法——CASA（Clustering Algorithm of SA coordination）。通过考虑事件发生频率和执行任务能耗影响，建立全网能耗模型，并以数据传输时延、连通度和通信半径为约束条件，确定簇的理想节点数，以此作为执行器节点部署的基础。同时确定簇内 SA 协作通信中的理想传输半径，自适应传感器节点拓扑结构，从而在保证实时性的前提下，保证能耗均衡。

6.3　CASA 算法原理与实现

6.3.1　参数定义与假设条件

1．参数定义（见表 6.1）

表 6.1　参数定义

Table 6.1　Parameters definition

名　　称	定　　义
E_{tx}	节点传输能耗
E_{rx}	节点接收能耗
l	一个数据包的长度

续表

名　称	定　义
α	发射能耗常数
β	发射电路（如 PLLs、VCOs）能耗
n	信道衰减倍数，取决于环境
E_{aggr}	节点数据汇集能耗
E_{sensing}	节点采集数据能耗
ξ	汇集能耗常数
χ	汇集后的包个数
γ	采集能耗常数
N_{sensor}	传感器节点数量
N_{actor}	执行器节点数量
d	执行器节点与基站/执行器节点通信范围，远大于传感器通信范围
T_{event}	事件发生频率/报告次数
E_{action}	单位时间执行操作能耗
t_{event}	执行操作时间
T_{com}	周期性采集数据的轮数
A	监控区域面积
R	传感器节点通信范围
R_{cluster}	簇半径
E_{sensor}	传感器节点能耗
E_{actor}	执行器节点能耗
E_{networks}	全网节点能耗

2．假设条件

传感器节点随机部署，单个节点的通信范围为圆面，且簇头位于簇中心。

6.3.2　基本能耗公式

传感器节点能耗主要由两部分组成：数据采集能耗、数据传输和接收能耗。执行器节点能耗包括三部分：接受本簇的转发包并融合计算所需能耗、融合后发送新包能耗、执行操作所需能耗。根据文献[6]发射硬件能耗模型，r 为节点通信距离，则传输和接收能耗公式为

$$E_{\text{tx}}(r,l)=(\alpha r^{n}+\beta)l \tag{6.1}$$

$$E_{\text{rx}}(l)=\beta l \tag{6.2}$$

式中，n 为信道衰减倍数，取决于应用环境。一般地，传感器节点取 n=2 时，α=10(pJ/bit)/m^2；由于执行器节点传输范围远大于传感器节点，参数选择也各

不相同，取 n=4 时，α'=0.0013(pJ/bit)/m^4，β=50nJ/bit。

数据汇集需要强大的计算能力，通常只有充当簇头的执行器节点具有此功能，当汇集 k 个 l 长度的数据报文时，其能耗公式为

$$E_{\text{aggr}}(k,l)=\xi\times k\times l \tag{6.3}$$

ξ为汇集能耗常数，典型值为 5(nJ/bit)/stream。相应的 k 个数据报文经过汇聚后转发的报文数由下式决定，其中 m 和 c 的取值见文献[7]：

$$\chi(k)=k\times m+c \tag{6.4}$$

传感器节点需要周期性采集环境数据，γ为采集能耗常数，典型值为 50 nJ/bit。则采集 l 长度的数据其能耗为

$$E_{\text{sensing}}(l)=\gamma l \tag{6.5}$$

6.3.3　优化模型建立

根据无线传感器/执行器网络 SA 协作的特点，本书采用执行器节点充当簇头的办法，减少由于利用传感器节点充当簇头，簇头选举造成的系统开销，以及避免选举过程中无法采集和传输传感器数据的弊端。因此，网络为一次成簇，当传感器节点随机部署后，通过从全网能耗的角度计算执行器节点理想数量，并由此确定簇的范围，簇头向各传感器节点发出邀请，收到邀请的传感器节点根据信号强弱加入簇。没有收到任何入簇邀请的传感器节点，发出入簇请求，根据响应信号的强弱入簇。

1．网络能耗模型建立

网络能耗主要由两部分组成：执行器节点能耗和传感器节点能耗。假定网络结构为周期采集型和事件驱动型两种的混合型。

则一个执行器节点的总能耗如下：

$$E_{\text{actor}}=T_{\text{com}}\left(E_{\text{rx}}\left(\frac{N_{\text{sensor}}\times l}{N_{\text{actor}}}\right)+E_{\text{aggr}}\left(\frac{N_{\text{sensor}}}{N_{\text{actor}}},l\right)+\chi\left(\frac{N_{\text{sensor}}}{N_{\text{actor}}}\right)E_{\text{tx}}(d,l)\right)+T_{\text{event}}E_{\text{action}}(t)t_{\text{event}} \tag{6.6}$$

式中，

$$E_{\text{rx}}\left(\frac{N_{\text{sensor}}\times l}{N_{\text{actor}}}\right)=\frac{\beta\times N_{\text{sensor}}\times l}{N_{\text{actor}}}$$

$$E_{\text{aggr}}\left(\frac{N_{\text{sensor}}}{N_{\text{actor}}},l\right)=\frac{\xi\times N_{\text{sensor}}\times l}{N_{\text{actor}}}$$

$$\chi\left(\frac{N_{\text{sensor}}}{N_{\text{actor}}}\right)=\frac{N_{\text{sensor}}\times m}{N_{\text{actor}}}+c$$

$$E_{\text{tx}}(d,l)=(\alpha' d^{n'}+\beta')l$$

为了方便讨论传感器节点通信能耗，我们将簇划分为若干同心圆，相邻圆半径之差为 R，如图 6.1 所示。

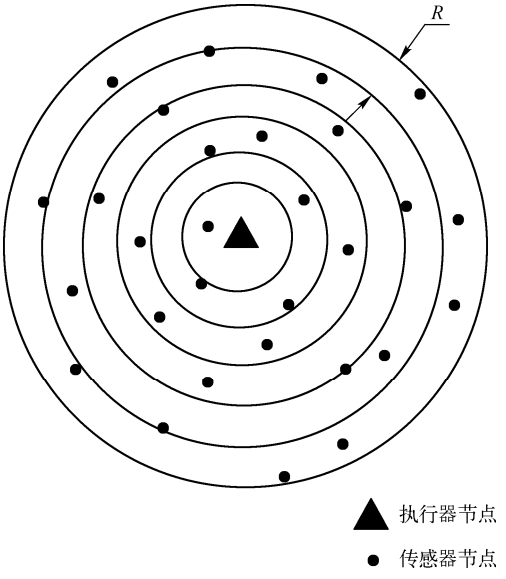


图 6.1　簇内多跳模式

Fig 6.1　Multi-hop mode in a cluster

假设执行器节点接收一个以 R 为通信半径传输的传感器数据，经过的跳数为 i，有 $1\leqslant i\leqslant R_{\text{cluster}}/R$。由于传感器节点并不是均匀分布，计算在第 i 轮转发时所在圆的平均节点数应为 $N_{\text{sensor}}(\pi R_{\text{cluster}}^2-\pi(iR)^2)/\pi R_{\text{cluster}}^2$，因此有 $N_{\text{sensor}}(\pi R_{\text{cluster}}^2-\pi(iR)^2)/\pi R_{\text{cluster}}^2$ 个数据包需要转发到第 i–1 轮，则在第 i 轮中，一个传感器节点需要转发的包个数为

$$\begin{aligned}k_i&=\frac{N_{\text{sensor}}(\pi R_{\text{cluster}}^2-\pi(iR)^2)/\pi R_{\text{cluster}}^2}{N_{\text{sensor}}(\pi(iR)^2-\pi((i-1)R)^2)/\pi R_{\text{cluster}}^2}\\&=\frac{R_{\text{cluster}}^2-i^2R^2}{R^2(2i-1)}\qquad\left(1\leqslant i\leqslant\frac{R_{\text{cluster}}}{R}\right)\end{aligned}\tag{6.7}$$

则第 i 轮转发时，传感器节点能耗应该包括接收和转发 k_{i} 个数据包的能耗，以及自身采集数据、发送数据能耗：

$$E_{\text{sensor}}=T_{\text{com}}(k_{\text{i}}E_{\text{rx}}(l)+(k_{\text{i}}+1)E_{\text{tx}}(R,l)+E_{\text{sense}}(l))\tag{6.8}$$

由图 6.1 可知，越靠近执行器节点，中转量越大，当 i=1 时，节点有最大中转载荷，因此要求节点必须具有最坏情况下转发数据的能量，有

$$E_{\text{sensor}} = T_{\text{com}}\left(\left(\frac{R_{\text{cluster}}^2}{R^2}-1\right)E_{\text{rx}}(l)+\frac{R_{\text{cluster}}^2}{R^2}E_{\text{tx}}(R,l)+E_{\text{sense}}(l)\right) \tag{6.9}$$

又因为簇半径为

$$R_{\text{cluster}} = \sqrt{\frac{A}{\pi N_{\text{actor}}}}$$

所以有

$$E_{\text{sensor}} = T_{\text{com}}l\left(\frac{A(2\beta+\alpha R^n)}{\pi N_{\text{actor}}R^2}-\beta+\gamma\right) \tag{6.10}$$

最后，得到全网节点中能耗为

$$E_{\text{networks}} = N_{\text{sensor}}E_{\text{sensor}} + N_{\text{actor}}E_{\text{actor}} \tag{6.11}$$

代入整理得

$$\begin{aligned} &E_{\text{networks}}(R,N_{\text{actor}}) = T_{\text{com}}lN_{\text{sensor}}[\gamma+\xi+(\alpha' d^{n'}+\beta')m]+\\ &[T_{\text{com}}lc(\alpha' d^{n'}+\beta')+T_{\text{event}}E_{\text{actiont}}(t)t_{\text{event}}]N_{\text{actor}}+\\ &\frac{N_{\text{sensor}}T_{\text{com}}lA(2\beta+\alpha R^n)}{\pi N_{\text{actor}}R^2} \end{aligned} \tag{6.12}$$

全网能耗公式可以表示成关于 R 和 N_{actor} 两个参数的方程，用于求解理想执行器节点数量和簇内理想通信半径。

2．簇内 SA 协作中的理想通信半径

簇内传感器节点与执行器节点通信方式可采用单跳和多跳方式：单跳方式能保证实时性，但耗能较多，特别是离执行器节点越远的传感器节点，消耗的能量越多；采用多跳方式，能耗分布有所不同，离执行器节点越近的传感器节点，消耗在汇集数据部分的能量越多。采用单跳或者多跳通信方式不能一概而论，需要根据具体应用环境参数进行分析。本书考虑传感器节点传输数据能耗和执行器节点执行任务能耗，建立网络能耗优化目标函数，并将时延和连通度作为约束条件，找出理想的执行器节点个数和传感器节点通信半径，采取适当的通信跳数，使得网络在满足实时性要求的同时能耗最小，延长网络生命期。

3．时延约束

一般地：传感器节点转发一组报文的时间由接收包时间 $t_{\text{rx}}(l)$、排队等待时间 $t_{\text{que}}(l)$、发送包时间 $t_{\text{tx}}(l)$组成，i 表示从传感器节点到执行器节点需要的跳数，则单个事件传输所需的总时间近似线性方程为

$$t_{i\text{-hop}}(l) = i(t_{\text{rx}}(l)+t_{\text{que}}(l)+t_{\text{tx}}(l)) \tag{6.13}$$

整理得

$$t_{i\text{-hop}}(R,N_{\text{actor}})=\frac{\sqrt{A}}{\sqrt{\pi N_{\text{actor}}}\times R}(t_{\text{rx}}(l)+t_{\text{que}}(l)+t_{\text{tx}}(l)) \tag{6.14}$$

为了保证网络的实时性，我们规定事件的报告时间不能超过一定时限：

$$t_{i\text{-hop}}(l)\leqslant t_{\text{deadline}} \tag{6.15}$$

则传感器节点的通信范围：

$$R\geqslant\frac{\sqrt{A}}{\sqrt{\pi N_{\text{actor}}}\times t_{\text{deadline}}}(t_{\text{rx}}(l)+t_{\text{que}}(l)+t_{\text{tx}}(l)) \tag{6.16}$$

$$\Rightarrow g_1(x)=\frac{A}{\pi t_{\text{deadline}}^2}(t_{\text{rx}}(l)+t_{\text{que}}(l)+t_{\text{tx}}(l))^2-R^2N_{\text{actor}}\leqslant 0 \tag{6.17}$$

4．连通度约束

根据文献，如果要达到 1-θ的连通概率，则相应的通信半径下限满足下列不等式：

$$R\geqslant\sqrt{\frac{A}{N_{\text{sensor}}}\ln\left(\frac{N_{\text{sensor}}}{\theta}\right)} \tag{6.18}$$

由此得到约束条件：

$$g_2(x)=\sqrt{\frac{A}{N_{\text{sensor}}}\ln\left(\frac{N_{\text{sensor}}}{\theta}\right)}-R\leqslant 0 \tag{6.19}$$

5．簇半径约束

由于传感器节点只与本簇节点通信，故其通信半径上限不能超过簇半径：

$$R\leqslant\sqrt{\frac{A}{\pi N_{\text{actor}}}}\Rightarrow\pi N_{\text{actor}}R^2\leqslant A \tag{6.20}$$

由此得到通信半径上限约束条件：

$$g_3(x)=\pi N_{\text{actor}}R^2-A\leqslant 0 \tag{6.21}$$

6．问题优化

令 $x=(R,N_{\text{actor}})$，以全网能耗作为目标优化函数，将传感器节点向执行器节点报告事件的时延和节点间连通度为约束条件，得到优化目标方程如下：

$$\text{最小化：}\ E_{\text{networks}}(x) \tag{6.22}$$

$$\text{约束条件：}\ g_1(x)=\frac{A}{\pi t_{\text{deadline}}^2}(t_{\text{rx}}(l)+t_{\text{que}}(l)+t_{\text{tx}}(l))^2-R^2N_{\text{actor}}\leqslant 0$$

$$g_2(x)=\sqrt{\frac{A}{N_{\text{sensor}}}\ln\left(\frac{N_{\text{sensor}}}{\theta}\right)}-R\leqslant 0$$

$$g_3(x)=\pi N_{\text{actor}}R^2-A\leqslant 0$$

6.3.4　优化模型求解

式（6.22）是一个标准的非线性优化问题，可以通过 Karush-Kuhn-Tucker（KKT）条件作为一阶必要条件，求得局部最优解。注意到约束条件 $g_3(x)=0$ 时，有 $\pi N_{\text{actor}}R^2=A$，表示簇内传感器节点的通信半径等于簇半径，通信模式即是单跳模式，此时的能耗目标函数变为

$$\begin{aligned}&E_{\text{networks-s}}(N_{\text{actor}})=T_{\text{com}}lN_{\text{sensor}}[\gamma+\xi+(\alpha' d^{n'}+\beta')m]+\\&[T_{\text{com}}lc(\alpha' d^{n'}+\beta')+T_{\text{event}}E_{\text{actiont}}(t)t_{\text{event}}]N_{\text{actor}}+\\&N_{\text{sensor}}T_{\text{com}}l\left(2\beta+\alpha\left(\frac{A}{\pi N_{\text{actor}}}\right)^{\frac{n}{2}}\right)\end{aligned}\tag{6.23}$$

欲求 N_{actor} 的最小值，只需令单跳能耗函数的一阶导数为 0，即

$$\frac{\mathrm{d}}{\mathrm{d}N_{\text{actor}}}E_{\text{networks-s}}(N_{\text{actor}})=0\tag{6.24}$$

$$\Rightarrow T_{\text{com}}lc(\alpha' d^{n'}+\beta')+T_{\text{event}}E_{\text{actiont}}(t)t_{\text{event}}+\frac{N_{\text{sensor}}T_{\text{com}}lA^{\frac{n}{2}}\alpha}{\pi^{\frac{n}{2}}}\cdot\left(-\frac{n}{2}\cdot N_{\text{actor}}^{-\frac{n}{2}-1}\right)=0$$

$$\Rightarrow N_{\text{actor}}=\left(\frac{nN_{\text{sensor}}T_{\text{com}}lA^{\frac{n}{2}}\alpha\pi^{-\frac{n}{2}}}{2(T_{\text{com}}lc(\alpha' d^{n'}+\beta')+T_{\text{event}}E_{\text{actiont}}(t)t_{\text{event}})}\right)^{\frac{2}{n+2}}\tag{6.25}$$

由于 $E_{\text{networks-s}}(N_{\text{actor}})$ 的二阶导数为正，所以上式求得的最小执行器节点数量 N_{actor} 即是全局最优解。

如果约束条件 $g_3(x)<0$，由 KKT 条件得到

$$\nabla E_{\text{networks}}(x)+\lambda_1\nabla g_1(x)+\lambda_2\nabla g_2(x)+\lambda_3\nabla g_3(x)=0\tag{6.26}$$

$$g_1(x)\leqslant 0\tag{6.27}$$

$$g_2(x)\leqslant 0\tag{6.28}$$

$$g_3(x)<0\tag{6.29}$$

$$\lambda_1\geqslant 0、\ \lambda_2\geqslant 0、\ \lambda_3\geqslant 0\tag{6.30}$$

$$\lambda_1 g_1(x)=0\tag{6.31}$$

$$\lambda_2 g_2(x)=0\tag{6.32}$$

$$\lambda_3 g_3(x)=0\tag{6.33}$$

式中，λ_1、λ_2、λ_3 为 KKT 条件的常数。为分析方便，将优化公式做如下改写，令

$$B = T_{\text{com}} l N_{\text{sensor}} [\gamma + \xi + (\alpha' d^{n'} + \beta') m] \tag{6.34}$$

$$C = T_{\text{com}} lc(\alpha' d^{n'} + \beta') + T_{\text{event}} E_{\text{actiont}}(t) t_{\text{event}} \tag{6.35}$$

$$D(R) = \frac{N_{\text{sensor}} T_{\text{com}} lA(2\beta + \alpha R^n)}{\pi R^2} \tag{6.36}$$

$$F = \frac{A}{\pi t_{\text{deadline}}^2} (t_{\text{rx}}(l) + t_{\text{que}}(l) + t_{\text{tx}}(l))^2 \tag{6.37}$$

$$G = \sqrt{\frac{A}{N_{\text{sensor}}} \ln\left(\frac{N_{\text{sensor}}}{\theta}\right)} \tag{6.38}$$

则有

$$\nabla E_{\text{networks}}(x) = \begin{bmatrix} D'(R)/N_{\text{actor}} \\ C - D(R)/N_{\text{actor}}^2 \end{bmatrix} \tag{6.39}$$

$$\nabla g_1(x) = \begin{bmatrix} -2N_{\text{actor}} R \\ -R^2 \end{bmatrix},\ \nabla g_2(x) = \begin{bmatrix} -1 \\ 0 \end{bmatrix},\ \nabla g_3(x) = \begin{bmatrix} 2\pi N_{\text{actor}} R \\ \pi R^2 \end{bmatrix}$$

分下列几种情况讨论：

（1）$\lambda_1 = \lambda_2 = \lambda_3 = 0$ 或者 $\lambda_2 = 0$ 且 $\pi\lambda_1 = \lambda_3 \neq 0$，有

$$\begin{cases} D'(R) = 0 \\ C - D(R)/N_{\text{actor}}^2 = 0 \end{cases} \tag{6.40}$$

解得

$$R = \left(\frac{4\beta}{(n-\alpha)\alpha}\right)^{\frac{1}{n}}$$

$$N_{\text{actor}} = \left(\frac{D(R)}{C}\right)^{\frac{1}{2}} = \left(\frac{N_{\text{sensor}} T_{\text{com}} lA(2\beta + \alpha R^n)}{\pi R^2 (T_{\text{com}} lc(\alpha' d^{n'} + \beta') + T_{\text{event}} E_{\text{actiont}}(t) t_{\text{event}})}\right)^{\frac{1}{2}}$$

同时，所求得的局部最优值必须满足约束条件，即

$$\frac{F}{R^2} \leqslant N_{\text{actor}} < \frac{A}{\pi R^2} \text{ 且 } R \geqslant G$$

（2）$\lambda_1 = \lambda_3 = 0$ 且 $\lambda_2 \neq 0$，或者 $\pi\lambda_1 = \lambda_3 \neq 0$ 且 $\lambda_2 \neq 0$，有

$$\begin{cases} R = G \\ \nabla E_{\text{networks}}(x) + \lambda_2 \nabla g_2(x) = 0 \end{cases} \tag{6.41}$$

解得

$$N_{\text{actor}} = \left(\frac{D(R)}{C}\right)^{\frac{1}{2}} = \left(\frac{N_{\text{sensor}} T_{\text{com}} lA(2\beta + \alpha G^n)}{\pi G^2 (T_{\text{com}} lc(\alpha' d^{n'} + \beta') + T_{\text{event}} E_{\text{actiont}}(t) t_{\text{event}})}\right)^{\frac{1}{2}}$$

同时，所求得的局部最优值必须满足约束条件，即

$$\frac{F}{R_2} \leqslant N_{\text{actor}} < \frac{A}{\pi R^2}$$

又因为

$\nabla E_{\text{networks}}(x)+\lambda_2\nabla g_2(x)=0$，且 $\lambda_2=0$，有

$$\lambda_2=\frac{D'(R)}{N_{\text{actor}}}>0\Rightarrow(n-2)\alpha R^n-4\beta>0$$

（3）$\lambda_3\neq0$ 即 $g_3(x)=0$，簇内通信转化为单跳通信模式，上面已作讨论，不再赘述。

（4）$\lambda_1\neq0$ 且 $\lambda_2=\lambda_3=0$，有

$$\begin{cases}F=R^2N_{\text{actor}}\\ D'(R)=2\lambda_1N_{\text{actor}}^2R\\ C=\dfrac{D(R)}{N_{\text{actor}}^2}+\lambda_1R^2\end{cases}\tag{6.42}$$

此方程组得到关于 R 的高次方程，目前的计算手段难以求解：同时，λ_1 的取值大小取决于相关参数，无法保证满足λ_1>0 的条件。

（5）$\lambda_1\neq0$、$\lambda_2\neq0$ 且 $\lambda_3=0$，有

$$\begin{cases}F=R^2N_{\text{actor}}\\ G=R\\ C=\dfrac{D(R)}{N_{\text{actor}}^2}-R^2\lambda_1=0\\ \dfrac{D'(R)}{N_{\text{actor}}}-2N_{\text{actor}}R\lambda_1-\lambda_2=0\end{cases}\tag{6.43}$$

解得

$$R=G;\quad N_{\text{actor}}=\frac{F}{G^2}$$

式中，λ_1 和λ_2 的取值大小取决于相关参数，无法保证满足λ_1>0 且λ_2>0 的条件，故不是可行解。

下面验证二阶充分条件，考虑集合

$$M=\{\boldsymbol{d}\mid\boldsymbol{d}\neq0,\nabla g(x)^{\mathrm{T}}\boldsymbol{d}=0\}$$

对于 Lagrange 函数在（1）、（2）两种情况下求得的 R、N_{actor} 处的 Hesse 矩阵 $\nabla^2L(x,\lambda)$，当 $\boldsymbol{d}\in M$，有

$$\boldsymbol{d}^{\mathrm{T}}\nabla^2L(x,\lambda)d>0\tag{6.44}$$

因此，（1）、（2）两种情况下求得的 R、N_{actor} 是局部解。

综上所述，由式（6.23）、式（6.40）、式（6.41）求出的三组解的值主要由所选参数决定，选择标准是：首先选择可行解，即求出的值必须符合约束条件；多个可行解的比较，可代回目标函数，能耗最少的为最优解。

6.3.5 CASA 算法实现

本算法分为部署阶段、分簇阶段：

1．部署阶段

部署包括传感器节点部署和执行器节点部署两部分。传感器节点可利用人工或者飞机随机部署在监测范围内；执行器节点的部署，本书假定执行器节点本身具有移动能力，并能够获取自身地理位置，借助虚拟力算法实现其部署。

虚拟力算法（Virtual Force Algorithm，VFA）[8]假设执行器节点、障碍物和监控区域边界均可对执行器节点施加引力或斥力。各执行器节点之间的作用力既有引力又有斥力。同时，各执行器节点又会受到边界和障碍物的斥力，根据其所受合力的大小和方向移动，直至达到目标位置或可移动距离的上限。

（1）虚拟力

① 相邻执行器节点之间的作用力。

VFA 算法利用距离来控制作用力大小，相邻执行器节点作用力包括执行器节点 i 与执行器节点 j 之间靠近时相互排斥力和濒临分离时相互吸引力。R_{A} 表示理想执行器节点个数下作用半径，由 $R_{\mathrm{A}}=\sqrt{\dfrac{A}{\pi N_{\mathrm{actor}}}}$ 公式获得，d_{ij} 表示执行器节点 i 与执行器节点 j 之间的距离，则有[93]

$$F_{ij}=\begin{cases}0 & d_{ij}=R_{\mathrm{A}}\\ (\omega_{\mathrm{A}}(1/R_{\mathrm{A}}-1/d_{ij}),\theta_{ij}) & d_{ij}>R_{\mathrm{A}}\\ (\omega_{\mathrm{R}}(R_{\mathrm{A}}-d_{ij}),\theta_{ij}) & d_{ij}<R_{\mathrm{A}}\end{cases} \tag{6.45}$$

式中，θ_{ij} 为以执行器节点 i 为起点指向执行器节点 j 的向量与横坐标正方向的夹角；ω_{A} 和 ω_{R} 为执行器节点间的引力系数和斥力系数。

② 障碍物斥力。

为了防止移动执行器节点在自动展开过程中碰撞障碍物，可设置障碍物指示点，该指示点对靠近的执行器节点会产生斥力，从而避免发生碰撞。执行器节点 i 受到障碍物指示点 k 的作用力可表示为

$$F_{ik}=\begin{cases}\dfrac{\eta m_i m_k}{d_{ik}^{\varphi}} & 0<d_{ik}\leqslant L\\ 0 & d_{ik}>L\end{cases} \tag{6.46}$$

式中，η、φ为增益系数；L 为执行器节点受到距障碍物斥力作用的距离；d_{ik} 为执行器节点 i 受到障碍物指示点 k 的距离。

③ 监控区域边界斥力。

$$F_{i\mathrm{B}}=\begin{cases}0 & d_{iq}\leqslant\sqrt{3}R_{\mathrm{A}}\\ \dfrac{\delta}{d_{iq}^{2}}\left(\dfrac{x_i-q}{d_{iq}}\right) & d_{iq}>\sqrt{3}R_{\mathrm{A}}\end{cases}\tag{6.47}$$

式中，δ为正常数；q 为执行器节点 i 到距离它最近的目标区域边界线的垂直投影点的位置。则执行器节点 i 所受到的合力为

$$F_i=\sum_{j=1,j\neq i}^{N_{\mathrm{actor}}}F_{ij}+\sum_{k=1,k\neq i}^{N_{\mathrm{obstacle}}}F_{ij}+F_{i\mathrm{B}}\tag{6.48}$$

在横、纵坐标方向上的分力大小分别是

$$F_{ix}=\sum_{i\in N_{\mathrm{actor}}}(|F_{ij}|\times\cos\theta_{ij})$$

$$F_{iy}=\sum_{i\in N_{\mathrm{actor}}}(|F_{ij}|\times\sin\theta_{ij})$$

（2）执行器节点的运动公式

最终，执行器节点 i 就在合力作用下进行移动，其原位置坐标（x_{old}, y_{old}）被更新为新位置坐标（x_{new}, y_{new}）：

$$x_{\mathrm{new}}=\begin{cases}x_{\mathrm{old}} & |F_i|=0\\ x_{\mathrm{old}}+\dfrac{F_{ix}}{|F_i|}\times\mathrm{MaxStep}\times\mathrm{e}^{\frac{-1}{|F_i|}} & |F_i|>0\end{cases}\tag{6.49}$$

$$y_{\mathrm{new}}=\begin{cases}y_{\mathrm{old}} & |F_i|=0\\ y_{\mathrm{old}}+\dfrac{F_{iy}}{|F_i|}\times\mathrm{MaxStep}\times\mathrm{e}^{\frac{-1}{|F_i|}} & |F_i|>0\end{cases}\tag{6.50}$$

式中，MaxStep 是执行器节点最大移动距离。

（3）执行器部署算法描述

```
/*VFA 算法伪代码*/
1：初始化阶段
   设置执行器作用力相关参数
   set loop = 0 ;              //设置循环变量;
   set loop_max ;              //设置最大循环次数;
2：循环阶段
While ( loop < loop_max) do
{
   读入执行器的当前位置坐标（x_old, y_old）;
   Send_request(loop) ;        //在通信半径 R_A 范围内对其他所有执行器发送位置请求;
   Receive_response ;          //接收在 R_A 范围内的所有执行器的回应;
```

```
        根据式（6.45）计算 F_ij;
        Detection ;                    //在感知半径范围内探测所有障碍物指示点和边界;
        根据式（6.46)和式（6.47）计算 F_ik 和 F_iB;
        根据式（6.48）计算合力 F_i;
        if(F_i==0)
        {
            break;
        }
        else
        {
            根据式（6.49)、式（6.50）计算移动节点移动到新位置坐标（x_new, y_new);
            执行器向新位置坐标进行移动;
        }
        loop = loop + 1 ;
    }
```

2．分簇阶段

一旦所有节点部署完成，执行器节点作为簇头，向网络其他传感器节点发出邀请，收到邀请的传感器节点根据信号强弱，选择信号强的簇头发出应答，簇头建立加入该簇的节点信息列表，一定时限后，发出指令，终止分簇阶段，转入工作阶段。

分簇阶段各信号及发射功率半径见表 6.2。

表 6.2　信号及功率半径

Table 6.2　Signal types and sending power

信 号 名 称	发送功率半径	意　　义
CHDs	R_{cluster}	簇头申明信号
cluster search	R_{cluster}	请求加入本簇信号
joinCH	R_{cluster}	接受并加入本簇信号
Ack	R_{cluster}	同意入簇请求

算法流程：

步骤 1：初始化执行器节点，自动成为簇头，每个簇头拥有一个 ID。

步骤 2：簇头以发射功率半径 R_{cluster} 发出申明信号 CHDs，寻找簇成员。

步骤 3：节点在一定时限内收到多个 CHDs 信号，如果节点自身为簇头，不作反应；如果节点是传感器节点，则根据信号强弱，选择最强信号的 ID 加入，向该簇头发出 joinCH。

步骤 4：如果某传感器节点在一定时限内没收到任何簇申明信号，就主

动发出簇找寻信号 cluster search，在其他簇头发回的响应信号中选择能量最强的分簇加入，向该簇头发出 joinCH，转入步骤 6。如果超时没有收到任何响应信号，再重发 cluster search：超过重发次数，则说明簇头已死，停止动作。

步骤 5：簇头收到多个 joinCH 信号后，建立簇成员信息表，发出同意信号 Ack，并将簇内通信理想半径 R 发给本簇成员；簇头收到 cluster search 信号，检查分簇成员是否达到最大成员值 M，如果没有，就发出 CHDs 加入邀请；否则，不响应。

步骤 6：一旦簇成员收到同意信号 Ack，就确定为该簇成员，只与簇头进行业务通信，转入工作阶段，不再响应 CHDs 信号。

6.4　算法仿真与性能分析

为了验证 CASA 算法性能，本书利用 MATLAB 和 NS2 仿真平台，针对执行器节点理想数量、虚拟力算法部署、网络通信开销和网络性能等指标开展仿真实验。

6.4.1　执行器节点理想数量

该实验主要测试在不同数量的传感器节点与执行器节点理想个数的关系，实验参数见表 6.3，其余参数取典型值将在 6.4.2 节提及。与文献[4]的 k-IDS 算法、文献[2]的 RCR 算法进行比较：

表 6.3　实验参数

表 6.3　Experiment parameters

参 数 名 称	取　　值	参 数 名 称	取　　值
事件发生频率 T_{event}	3 次/min	周期性采集数据的轮数 T_{com}	10^4
执行操作时间 t_{event}	5s	单位时间执行操作能耗 $E_{action}(t)$	5000nJ·s^{-1}
报文长度 l	512bit	连通概率 $1-\theta$	0.99
事件报告时限 $t_{deadline}$	200ms	接收包时间 $t_{rx}(l)$	5ms/包
排队等待时间 $t_{que}(l)$	15ms/包	发送包时间 $t_{tx}(l)$	5ms/包

由图 6.2 可知，执行器节点的总趋势是随着随机部署的传感器节点数量增加而增加，然而与其他两种算法比较，本书通过对全网络能耗进行建模优化，并以时延和连通度作为约束条件，求出理想执行器节点数量，得到的执行器节点数量值，相对更加精确合理。

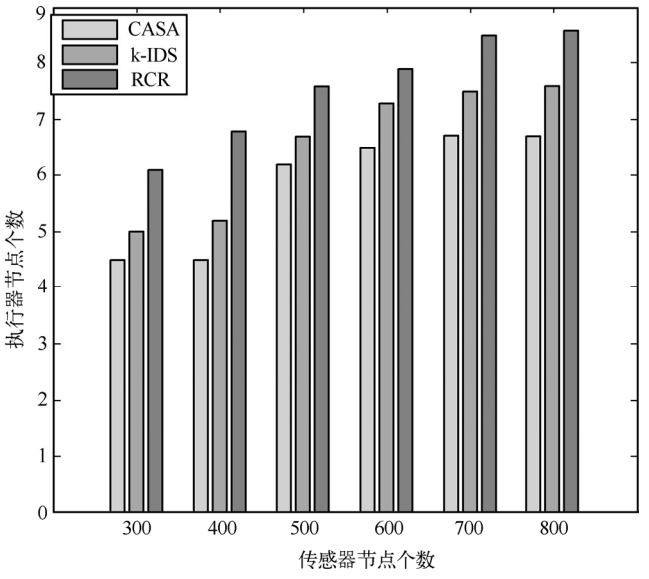
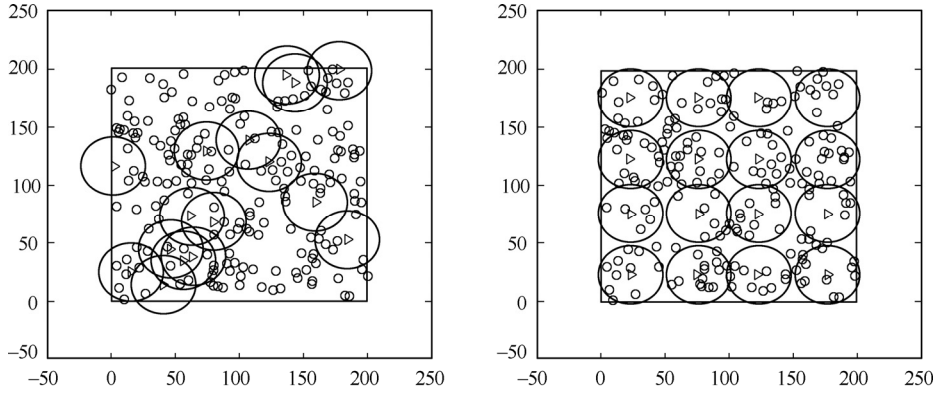


图 6.2　传感器节点个数-执行器节点个数关系图

Fig. 6.2　Relationships between sensors number and actors number

6.4.2　基于 VFA 算法的执行器节点部署实验

该实验主要检验基于局部虚拟力的执行器节点部署算法。目标区域是 200m×200m 的方形区域，由 16 个移动执行器节点和 200 个静止传感器节点随机部署在区域内，一起组成无线传感器/执行器网络。图 6.3 显示执行器节点随机部署的初始覆盖情况，然后执行器节点开始运行基于虚拟力的部署算法。实



（a）执行器节点初始阶段　　（b）执行器节点部署完成

图 6.3　基于 VFA 算法的执行器节点部署图

Fig.6.3　Deployment of actors after (a) initial random placement and after the optimization of the (b) VF algorithm.

验参数：执行器节点作用力半径 R_A=25m，ω_A=1；ω_R=5；MaxStep=0.4R_A；设定目标位置相关的概率检测参数同文献[94]。

由图 6.3 可知，执行器节点在多次迭代执行虚拟力算法之后，能到达预定目标位置，形成均匀覆盖，完成部署。

6.4.3　算法通信开销

CASA 算法通信开销主要分为两个阶段。由于部署阶段采用 VFA 算法，与一般的部署算法没有太大差别，通信开销基本一致，故不做讨论。重点集中讨论分簇阶段的通信开销，主要集中在三个部分：

（1）簇头申明阶段，执行器节点以功率半径 $R_{cluster}$ 发出申明信号 CHDs

在执行器节点部署完成后，需要进行一次单向簇头申明，寻找簇成员。通信次数 $C_{CHDs}=N_{actor}$。

（2）簇成员收到申明信号后，以功率半径 $R_{cluster}$ 发出 joinCH 信号进行应答

如果一个簇成员收到多个簇头的申明信号，只选择一个进行应答，也就是说一个传感器节点只应答一次，则通信次数 $C_{joinCH}=N_{sensor}$。

（3）簇头发出同意信号 Ack

簇头收到多个传感器节点的 joinCH 信号，建立簇成员列表，并发出 Ack 同意信号，信号有多个簇成员 ID 列表，收到同意信号的簇成员检查 ID 列表，与自身 ID 匹配即表明入簇请求已经得到同意，转入工作状态。通信次数 $C_{Ack}=N_{actor}$。

从上述分析可以看出，由于分簇阶段的所有通信都是以 $R_{cluster}$ 发送，节点并不转发收到的信号，相对于工作阶段必须经过若干次簇头轮换的分簇算法来讲，本算法一次成簇，通信开销很小。

6.4.4　网络性能

基于理想传感器节点传输范围和执行器节点个数部署的网络，本书从网络能耗和平均时延这两个主要指标检验算法性能。

图 6.4 显示为不同网络规模下，簇内传感器节点向执行器节点报告事件发生的平均时延。图中可知，随着网络规模的扩大，RCR 和 k-IDS 算法中传感器节点向执行器节点报告的时延也在增加，反映了对网络动态的变化，缺乏对应的调整。而经过优化后的网络，不同的规模有不同的传输半径，对网络规模的增加有相应的调整，保证了事件报告时延维持较低水平，波动也比较平稳。

为了比较算法能耗均衡问题，实验在 6.5.2 节的基础上，对于假设 200 个传感器节点组成的无线传感器网络部署在 200m×200m 区域，构造执行器节点的

剩余能量与初始能量比值作为能量均衡指数。

$$\mathrm{EB}_i = E_{i-\mathrm{rest}} / E_{i-\mathrm{init}} (1 \leqslant i \leqslant m)\ ,$$

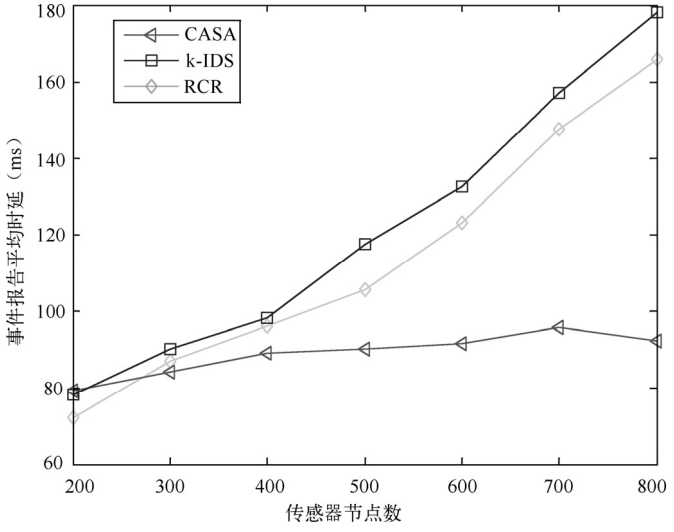


图 6.4　平均时延-网络节点关系图

Fig. 6.4　Relationships between average delay and sensors number

式中，m 为执行器节点总数。随着监测时间的推移，考察网络中执行器节点其对应的数学期望 E 和方差 V。

由图 6.5，各算法的执行器节点能耗均衡指数的数学期望差别并不大，但从图 6.6 可知，k-IDS 和 RCC 算法在事件频发的“热区”效应下，各执行器能量

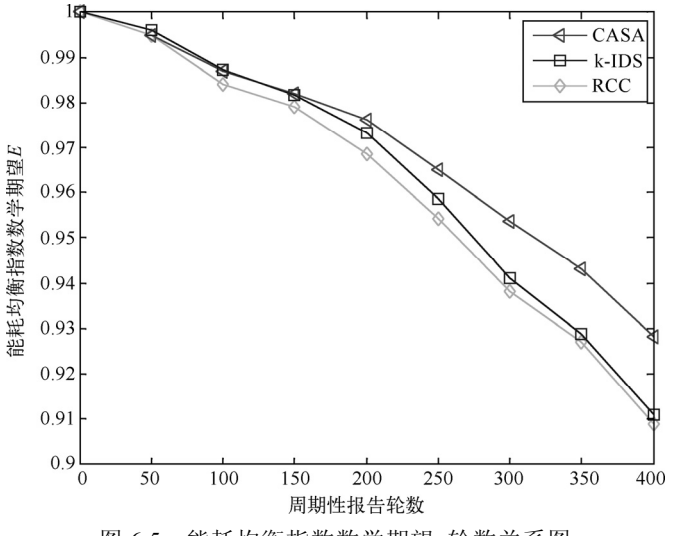


图 6.5　能耗均衡指数数学期望-轮数关系图

Fig6.5　Relationships between mathematical expectation of energy balance index and report cycles

分布差异明显，而在 CASA 策略下其方差波动值最小，反映了各执行器节点能耗比较平均，负载更加均衡。随着周期性报告轮次增加，这种趋势更加明显。

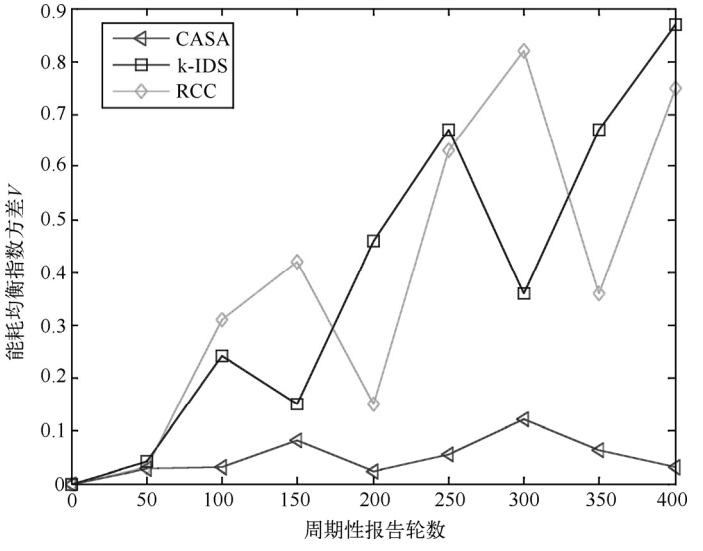


图 6.6　能耗均衡指数方差-轮数关系图

Fig6.6　Relationships between variance of energy balance index and report cycles

6.5　小结

针对无线传感器/执行器网络中传感器节点与执行器节点之间 SA 通信协作问题，提出一种基于簇内 SA 协作模型的分簇算法 CASA：策略综合考虑时延、连通度等约束条件，以网络能量优化为目标，建立非线性优化函数，求解网络理想执行器节点个数和传感器节点传输半径等网络部署所需参数，并在此基础上基于虚拟力算法完成节点部署。网络平均时延和生存期等仿真结果表明，相比典型算法，该策略能够在满足一定连通度的前提下，优化网络部署，增强网络实时性和能量均衡性。

参 考 文 献

[1] Li Fangmin. Real-time energy-aware cluster-based routing protocol for wireless sensor and actor networks[J]. Jisuanji Yanjiu yu Fazhan/Computer Research and Development. 2008,45(1):26-33.

[2] Shah G. A. Bozyigit M. Akan O. Real Time Coordination and Routing in Wireless Sensor

and Actuator Networks[C]. In Proceedings of the 6th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking NEW2AN. 2006:365-383.

[3] N. Trivedi G. Elangovan S. S. 'Ripples': A message-efficient, distributed clustering algorithm for wireless sensor and actor networks[C]. IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems. 2006:53-58.

[4] kkaya Kemal A. Clustering of wireless sensor and actor networks based on sensor distribution and connectivity[J]. Journal of Parallel and Distributed Computing. 2009, 69(6):573-587.

[5] K. Akkaya M. Younis Coverage and delay aware actor placement mechanisms for wireless sensor and actor networks[J]. International Journal of sensor networks. 2008. 3 (3):11-25.

[6] Rappaport. T. Wireless Communication Principles and Practice(2nd Edition) (M). Upper Saddle River, NJ. : London: Prentice Hall PTR. 2002.

[7] V Mhatre C. Rosenberg. Design guidelines for wireless sensor networks: communication, clustering and aggregation[J]. Ad Hoc Networks, Elsevier. 2004,2(1): 45-63.

[8] Zou Y Chakrabarty K. Sensor deployment and target localization based on virtual forces[A]. IEEE INFOCOM[C]. Piscataway,NJ,USA;IEEE Press. 2003. 1293-1303.

[9] Wang Xue. An Improved Co-evolutionary Particle Swarm Optimization for Wireless Sensor Networks with Dynamic Deployment[J]. Sensors. 2007,7(3): 354-370.

第 7 章　AA 实时协作框架

无线传感器/执行器网络（Wireless Sensor and Actor Networks, WSANs）是一种新型、自组织的无线异构网络，由大量传感器节点（Sensor）和少量执行器节点（Actor）组成。为完成各种分布式感知任务和执行任务，传感器节点与执行器节点（Sensor-Actor，SA）、执行器节点之间（Actor-Actor，AA）需要大量协作。SA 协作主要完成感知任务并报告事件，能耗较小；而 AA 协作负责任务执行，对网络能耗均衡影响相对较大。因此，AA 协作是 WSANs 关注的重点，要求执行器节点对事件快速响应，同时网络能耗均衡，避免事件频发区域执行器节点执行能耗过大，降低网络寿命[1]。

Tommaso Melodia [2]等人提出一种适用于无线传感器/执行器网络的分布式协作框架，将基于事件驱动的分簇结构中的 AA 协作问题转化为混合的整数非线性规划问题来解决，执行器节点之间谈判通过本地拍卖机制完成。但是每次只选出剩余能量最高的执行器节点执行任务，在事件频发场合，算法暴露出实时性不高的缺点。Shah G.A.[3]等人提出了实时协作与路由框架（RCR），并未对 AA 协作进行详细分析。Edith C. H. Ngai[4]等人提出一种实时通信框架，其协作机制依赖地理路由协议，使执行器节点很容易根据情况改变位置，快速响应事件，缺点是忽略了网络能耗均衡问题。文献[5]设计了一种基于移动 ad hoc 网络（MANETs）的自治防火系统，比较了受控机动性（Controlled Mobility）、预测技术（Predictive Techniques）、知识利用（Knowledge Exploitation）三类协作方法，其中比较前两种协作，知识利用技术扩大了执行器节点的移动自由度，显示出更好的网络性能，但是实时性和能耗均衡没有讨论。Kemal Akkaya 等人[6]提出一种新的分布式算法，用来优化网络拓扑结构，关注的重点在于连通度和最小化执行器节点移动距离问题。类似的工作是文献[7]，文中提出的 DARA 算法，在执行器节点通信网络中的链路失效后，通过协作多执行器节点重部署来重建通信链路。与文献[6]一样，协作的目的在于提高网络性能，没有对实时性和能耗均衡问题提出相关策略。

本书提出一种实时协作（RC）算法，主要针对事故频发区域执行器节点能耗过大的问题，将任务按照可分性和区域限制分解成一组任务执行单元，利用拍卖机制，根据网络情况进行动态范围任务招标，各执行器对拍卖任务元进行基于熵权的代价估算，拍卖执行器节点根据返回的代价值合理安排任务指派方案，使

得任务由多个执行器节点并发协作完成，保证网络能耗均衡，同时提高了实时性。

7.1 AA 协作模型

无线传感器/执行器网络中的 AA 协作，主要解决如何实现多执行器节点的任务分配问题。文献[1]将 AA 协作中的任务分为单执行器任务（Single-Actor Task，SAT）和多执行器任务（Multi-Actor Task，MAT）：SAT 指任务只需要一个执行器节点完成，不需要其他执行器节点协作。对于 SAT 任务，协作算法需要在多个执行器节点中挑选出一个“最好”执行器节点执行该任务。MAT 指任务可以分解，由多个执行器协作执行，协作算法必须建立任务执行评价体系，合理分配任务，以达到算法的性能目标。如本书 2.3 节介绍，协作算法分为集中式和分布式两大类。集中式算法按组织方式，由决策节点制定行动方案，其他执行器节点接收指令并执行；分布式算法允许执行器节点相互协调，针对网络动态变化采取行动。其中分布式体系结构相对于集中式结构，具有计算量小、通信量小、容错性强等优点，被广泛用于多执行器节点动态协作中。AA 协作算法面临如下研究议题：

（1）根据事件类型，采取相应的任务，并判断任务是 SAT 任务还是 MAT 任务；

（2）执行器节点之间建立通信模型，用于交换执行器节点信息；

（3）为达到算法性能目标（如实时性、能耗均衡等），确定 AA 协作的基本方案，找出执行任务的“最好”执行器节点或者节点集合。

本书的研究环境是以执行器节点为簇头的分簇网络结构如图 7.1 所示：当事件发生后，传感器节点向本簇执行器节点报告，该执行器节点根据事件属性执行相应的任务，或者自己执行，或者联合其他执行器节点，协同完成任务。

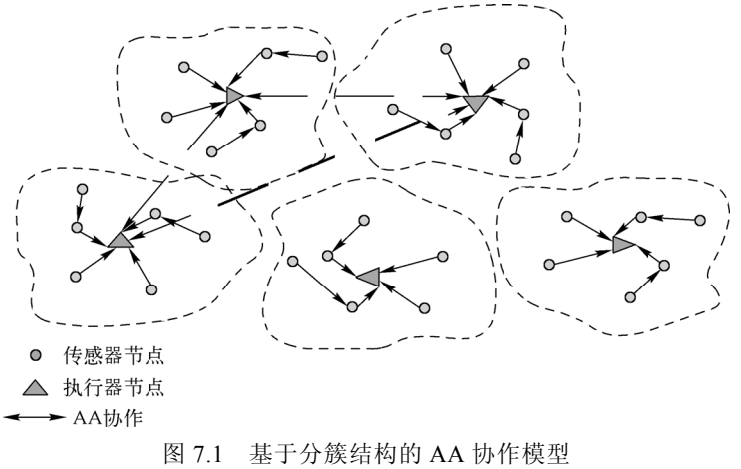


图 7.1　基于分簇结构的 AA 协作模型

Fig. 7.1　Actor-actor coordination model based on cluster

7.2　任务类型分解

任务由一个或者多个操作组成，这些操作可以由网络中的多个执行器节点协作完成，这样可以解决处于事件频发区域的执行器节点能量消耗过快，响应时间变长的问题，使得网络能耗均衡，延长网络寿命。假设每个执行器节点具有移动能力，条件相同。

【定义 7.1】完成一个特定目标的操作（包括通信、数据处理、执行动作等），执行时间和能耗可预知，称为一个任务执行单元，简称任务元，记为 t。

一个任务 $T=t_1 \cup t_2 \cup ... \cup t_n$，且 $\forall t_i, t_j \in T$，$\exists t_i \cap t_j = \varnothing$，称任务 T 具有可分性。

【定义 7.2】若一个任务不具有可分性，只能由单个执行器节点完成的任务，称为单执行器节点任务（Single-Actor Task，SAT）；若一个任务具有可分性，可以由多个执行器节点共同完成，则称为多执行器节点任务（Multi-Actor Task，MAT）。

【定义 7.3】执行器节点集合 $A=\{a_1, a_2, ..., a_m\}$，至少存在一个任务元必须由指定区域的执行器节点执行，即 $\exists t_i \to a_j$，且 $t_i \in T$，$a_i \in A$，称其为有区域限制（Special Region，SR）。否则，称为无区域限制（Free Region，FR）。

【定义 7.4】若 $O=\{o_1, o_2, ..., o_n\}$ 为任务 T 的 n 个任务元的执行顺序，o_1 只有一个后继任务元，o_n 只有一个前趋任务元，其余任务元拥有一个前趋和一个后继，称为其执行具有工序限制（Ordered Execution，OE）；否则，称为无工序限制（Free Execution，FE）。

执行器节点根据事件属性进行任务分解（见图 7.2），并判断任务类型以及相应处理见表 7.1。

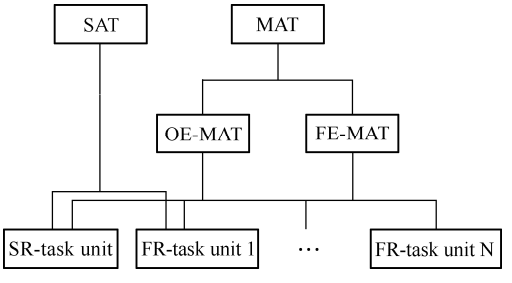


图 7.2　任务分解

Fig.7.2　Task decomposition

表 7.1　任务分类

Table 7.1　Task classification

任 务 类 型	执行器节点动作
有区域限制任务元（SR-task unit）	由该执行器节点执行
无区域限制任务元（FR-task unit）	进行拍卖，选出局部最佳执行器节点集合执行
无工序限制的多执行器节点任务（FE-MAT）	可分解成若干个 SR 任务元和 FR 任务元
有工序限制的多执行器节点任务（OE-MAT）	分解的任务元必须按工序执行，分派的评估与计算更加复杂，第 5、6 章专门讨论

7.3　基于拍卖机制的任务分派

当任务分解完成后，拍卖执行器节点通过拍卖机制把各任务元信息发给网络中的其他执行器节点。这里把一个执行器节点将任务分解成若干任务元，并发出拍卖邀请，称为拍卖执行器节点。拍卖执行器节点通过广播的形式发出邀请，包括需拍卖的任务元的能耗、事件位置等信息。其他执行器节点同意参与拍卖，并发回综合代价值进行投标，称为投标执行器节点。拍卖执行器节点根据收到的综合代价值，确定最佳任务分派方案。

7.3.1　基于事件的动态招标范围

为了选出合适执行器节点集合，候选节点数量应由两部分组成：基本候选节点和冗余候选节点。基本候选节点由任务分解后的单元数量决定，冗余候选节点数量由事件发生频率和候选节点不足导致的重新招标次数决定，由此可知，候选节点的招标范围随着事件不同而动态变化。

图 7.3 显示以拍卖执行器节点为中心的候选节点分布。冗余值的确定需要谨慎选择，冗余值太大，通信代价和计算成本过高；冗余值过小，不利于选出适当的执行器节点集合。冗余值受到两个因素影响：事件发生频率和重新招标次数。前者反映了当事件频繁发生时，该区域的执行器节点能耗相对更大，需要更大范围内寻找候选节点，以便维持网络能量均衡；后者是在初次招标候选节点不足的情况下，扩大招标范围，加大冗余值，以便找出合适的候选节点集合。重新招标次数越多，冗余值应越大。为分析简便，下面的执行器节点简称节点。

【定义 7.5】节点当前已发生事件数量与网络已发生事件总数量之比，称为事件发生频度。记为μ_1，有

$$\mu_1 = \frac{t_{\text{local-e}}}{t_{\text{all-e}}} \tag{7.1}$$

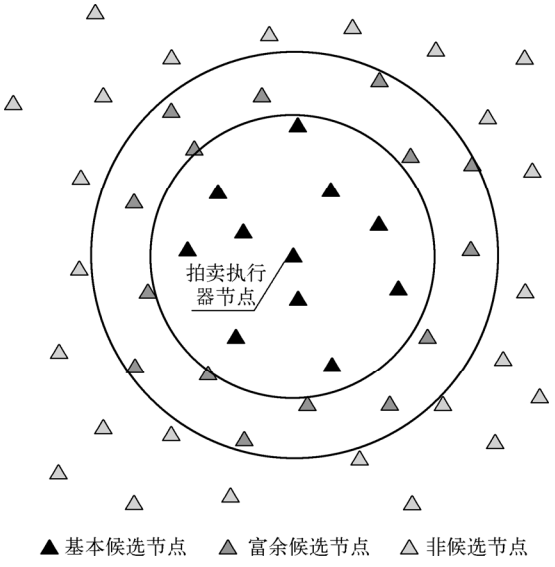


图 7.3　候选节点分布
Fig.7.3　Distribution of candidate nodes

式中，$t_{\text{local-e}}$ 为本地事件发生次数；$t_{\text{all-e}}$ 为全网已发生事件总数。

【定义 7.6】节点当前招标失败次数与最大允许失败次数之比，称为招标失败频度。记为μ_2，有

$$\mu_2 = \frac{t_{\text{bidding}}}{t_{\text{max-b}}} \tag{7.2}$$

式中，t_{bidding} 为当前招标失败次数；$t_{\text{max-b}}$ 为最大允许失败次数，考虑实时性，设值趋小。

则冗余因子λ为

$$\lambda = \omega_1\mu_1 + \omega_2\mu_2 \tag{7.3}$$

式中，

$$\omega_1 = \frac{\mu_1}{\mu_1 + \mu_2} \quad \omega_2 = \frac{\mu_2}{\mu_1 + \mu_2} \tag{7.4}$$

由式（7.3）、式（7.4）可知，首次招标时，冗余因子主要由事件发生频度决定，频度越大，冗余因子相应变大；当招标失败后，ω_2 权重增大，当 $t_{\text{max-b}}$ 设定较小时，$\omega_2>\omega_1$，反映出再次招标时，失败对冗余因子的取值影响较大。

上述分析只考虑了事件发生频度和招标失败频度对冗余因子的影响，由于网络的动态变化，最近历史招标经验和本次招标失败程度也应作为确定下次招标候选节点个数的依据。

设 $m_{\text{last-e}}$ 为上次招标的任务元数量，$N_{\text{last-c}}$ 为相应的招标成功时的候选节点数量，m 为本次招标的任务元数量，则本次招标的基本候选节点数 $N_{\text{base-c}}$ 有

$$N_{\text{base-c}} = \frac{N_{\text{last-c}}}{m_{\text{last-e}}} m \tag{7.5}$$

上式把最近招标成功的历史经验作为确定本次招标的基本候选节点数，有利于提高首次招标的成功率。再将式（7.3）的冗余因子引入式（7.5），基本可以确定本次招标的候选节点数。但是由于招标失败频度μ_2只考虑招标失败次数，并没有考虑失败的程度，因此在再次招标中，必须考虑首次招标中已参与招标的节点个数。

设 n 为已参与投标节点数，$N_{\text{candidate}}$ 为本次招标的候选节点数，则

$$N_{\text{candidate}} = (N_{\text{base-c}} - n)(1+\lambda) + n \tag{7.6}$$

将式（7.3）、式（7.5）代入式（7.6）得

$$N_{\text{candidate}} = \left(\frac{N_{\text{last-c}}}{m_{\text{last-e}}} \cdot m - n \right)(1 + \omega_1\mu_1 + \omega_2\mu_2) + n \tag{7.7}$$

首次招标时，n=0，式（7.7）变形

$$N_{\text{candidate}} = \left(\frac{N_{\text{last-c}}}{m_{\text{last-e}}} \cdot m \right)(1 + \omega_1\mu_1) \tag{7.8}$$

此时，候选节点数只与最近招标历史经验和事件发生频度有关，最近招标历史经验作为先验数据，确定本次招标的基本候选节点数，同时引入事件发生频度。当事件发生频度增加，容易形成热点区域，区域内执行器节点能耗过多，式（7.8）中的冗余因子$\lambda=\omega_1\ \mu_1$相应变大，使得更多的执行器节点参与和承担，达到网络能耗均衡。我们注意到；当拍卖执行器节点第一次招标，此时没有先验数据作为依据，候选节点数的准确性不够，影响首次招标的成功率。依据式（7.7）中的μ_2，候选节点数呈线性增加，一旦得到第一次招标成功的先验数据，以后的招标中的首次成功率将会大大提高。最坏的情况是招标次数达到最大允许失败次数，冗余因子λ=1，且已参与投标的候选节点数 n=0，此时式（7.7）变形为

$$N_{\text{candidate}} = 2 \cdot \frac{N_{\text{last-c}}}{m_{\text{last-e}}} \cdot m \tag{7.9}$$

综上所述，式（7.7）反映了影响招标的几个因素和对候选节点的贡献，特别是将最近历史招标候选节点数作为先验数据引入，大大增加了首次招标的成功率，保证了实时性。

候选节点数 $N_{\text{candidate}}$ 一旦确定，利用功率控制技术，相应的招标范围半径 R_{bidding} 可根据下列公式确定。

设执行器节点均匀部署，密度为σ，文献[96]证明了均匀分布的无线传感器网络在任意区域内的节点数目服从泊松分布，则在发射功率半径 r 范围内，与 $N_{\text{candidate}}$ 个节点连通的概率关系为

$$P\{N_{\text{candidate}}=k\}=\left(1-\sum_{l=0}^{k-1}\frac{(\rho\pi r^2)^l}{l!}\mathrm{e}^{-\rho\pi r^2}\right) \tag{7.10}$$

根据式（7.10），求解出半径 $R_{\text{bidding}}=r$。

又根据电波在自由空间传播损耗的 TwoRayGround 模型接收端功率[97]：

$$P_{\text{r}}=\frac{P_{\text{t}}G_{\text{t}}G_{\text{r}}h_{\text{t}}^2h_{\text{r}}^2}{Ld^4} \tag{7.11}$$

式中，P_{t} 为发送端天线发射功率；P_{r} 为接收功率；G_{t}、G_{r} 分别为发射、接收天线增益；h_{t}、h_{r} 分别为发送、接收天线高度；L 为天线长度。执行器节点通过侦听信道中的能量强度来判断是否有信号在信道中传输。当侦听到的能量强度高于节点的接收门限 P_{r} 时，接收节点能够正确接收信道内的数据。由此可得到执行器节点当前的发射功率：

$$P_{\text{t}}=\frac{P_{\text{r}}d^4L}{G_{\text{t}}G_{\text{r}}h_{\text{t}}^2h_{\text{r}}^2} \tag{7.12}$$

式中，$d=R_{\text{bidding}}$。

至此，完成了利用功率控制技术，实现基于事件的动态招标范围的推导。如果拍卖执行器节点首次招标由于候选节点不足导致失败后，在时限内可再次招标，利用功率控制技术扩大招标范围。一旦超过时限，为保证实时性，该任务交由本地执行。算法描述如下：

```
/*基于事件的动态招标策略伪代码*/
#define MAX_BIDDING_TIMES 2          //设最大允许招标失败次数
/*全局变量*/
Uint16_t   last_tu_number;           //设上次任务元数量
Uint16_t   last_candidater;          //设上次相应的候选节点数
/*招标函数*/
Uint16_t Bidd(Uint16_t tu_number)
{
  Uint16_t bidding_times;            //招标次数
  Uint16_t candidater;               //候选节点数
  Uint16_t bidder_number;            //参与投标节点数
  Uint16_t power;                    //功率半径
```

```
        typedef struct Bidder* bidder_inf;              //参与投标节点信息

        Uint16_t all_event_number = Get_all_event(); //得到全网以发生事件总数
        while(bidding_times < MAX_BIDDING_TIMES)
        {
            candidater = Compute_candidater(tu_number, bidder_number, bidding, last_
tu_number, last_candidater);                            //利用式（7.7）计算候选节点个数
            power = Compute_power(candidater);    //利用式（7.12）计算招标功率半径
            send((uint16_t *)&task_inf, power);
                            //以该功率半径进行招标，招标信息中包括任务信息
            delay(N);                                   //延时一段时间
            read_msg(bidder_inf);                       //读出参与投标的节点信息
            bidder_number = bidder_inf->bidder;    //读出参与投标节点数
            if(bidder_number >= tu_number)
                            //如果参与投标节点数大于或者等于任务单元数
           {
                last_tu_number = tu_number;       //记录此次任务元
               last_candidater = candidater;          //记录此次候选节点数
               break;
            }
            bidding_times++;                            //否则，继续招标
        }
        if(timer_fired)                             //超时，终止招标，任务交由本地执行
        {
            Location_do(task_inf);
        }
        else
        {
            Compute_arrany(bidder_inf, task_inf);   //未超时，进行任务分派
          }
      }
```

7.3.2 基于熵权的代价评估模型

收到邀请的空闲执行器节点根据邀请信息，计算执行各任务元的代价，发回参与投标的综合代价值，成为投标执行器节点。对于投标执行器节点对代价的评估，主要从三个指标进行，设第 i 个执行器节点完成第 j 个任务元的所需能耗、属性和事件发生坐标，建立执行器节点的能耗代价（E）、移动时延代价（A）、信用值（G）三个指标的代价函数 x_{ij}。

【定义 7.7】能耗代价指标 E：设执行器节点完成第 j 项任务元所需能量为

E_{action}，它与该执行器节点剩余能量 E_{residual} 的比值为

$$E=\frac{E_{\text{action}}}{E_{\text{residual}}} \tag{7.13}$$

称 E 为该执行器节点完成任务元 j 的能耗代价指标。

特别的，当 $E>1$ 时，$x_{ij}\to\infty$，表示该执行器节点剩余能量无法完成该任务元。

【定义 7.8】移动时延代价指标 A：已知事件发生位置 P 和执行器节点位置 S，则它们的距离为 $\text{dist}(P,S)$，有 $\text{dist}(P,S)=\|P-S\|_2$，其中 $\|.\|_2$ 指欧几里得距离。同时，该执行器节点的移动半径为 R_{a}，则有效执行代价指标：

$$A=\frac{\text{dist}(P,S)}{R_{\text{a}}} \tag{7.14}$$

特别的，当 $A>1$ 时，$x_{ij}\to\infty$，表示该任务超出该执行器节点的移动范围。

【定义 7.9】信用值 G：设执行器节点分派任务次数为 t_{assigned}，成功执行次数为 t_{success}，则信用值表明执行器节点在参与竞争中的可信任程度：

$$G=1-\frac{t_{\text{success}}}{t_{\text{assigned}}} \tag{7.15}$$

在多指标综合评价中，确定指标权重的方法主要有主观赋权法和客观赋权法。主观赋权法是一类根据评价者主观上对各指标的重视程度来决定权重的方法；客观赋权法所依据的赋权原始信息来源于客观环境，根据各指标的联系程度或各指标所提供的信息量来决定指标的权重。熵权法[98]属于客观赋权法，利用信息熵的概念，客观反映评价对象的动态变化。

设有 m 个任务元，n 项评价指标，形成原始指标数据矩阵 $\boldsymbol{X}=(x_{ij})_{m\times n}$，利用信息论中熵 $H(x)=-\sum_{i=1}^{n}p(x_i)\ln p(x_i)$ 作为系统无序程度的度量，对于某项指标 x_j，指标值 x_{ij} 的差距越大，则该指标在综合评价中所起的作用越大；如果某项指标的指标值全部相等，则该指标在综合评价中不起作用。所以可以根据各项指标值的变异程度，利用熵计算出各指标的权重，为多指标综合评价提供依据。

利用熵权法进行综合评价的步骤如下：

（1）建立评价矩阵 $\boldsymbol{X}$。

$$\boldsymbol{X}=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{7.16}$$

（2）将各指标同度量化，计算第 j 项指标下第 i 个任务元指标值的比重 p_{ij}。

$$p_{ij}=\frac{x_{ij}}{\sum_{i=1}^{m}x_{ij}} \tag{7.17}$$

则评价矩阵变为 P：

$$P=\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{bmatrix} \tag{7.18}$$

（3）计算第 j 项指标的熵值 e_j。

$$e_j=-k\sum_{i=1}^{m}p_{ij}\ln p_{ij} \tag{7.19}$$

式中，$k>0$，$e_j\geqslant 0$。如果 x_{ij} 对于指标 j 全部相等，有

$$p_{ij}=\frac{x_{ij}}{\sum_{i=1}^{m}x_{ij}}=\frac{1}{m} \tag{7.20}$$

此时 e_j 取极大值，即

$$e_j=-k\sum_{i=1}^{m}\frac{1}{m}\ln\frac{1}{m}=k\ln m \tag{7.21}$$

若设 $k=\ln\frac{1}{m}$，于是有 $0\leqslant e_j\leqslant 1$。

（4）计算第 j 项指标的差异性系数 g_j。

对于给定的指标 j，x_{ij} 的差异性越小，则 e_j 越大，当 x_{ij} 全部相等时，$e_j=e_{\max}=1$，此时对于方案的比较，指标 x_j 毫无贡献；当各方案的指标值相差越大时，e_j 越小，该项指标对于方案比较所起的作用越大。定义差异性系数：

$$g_j=1-e_j \tag{7.22}$$

（5）计算第 j 个指标的熵权。

$$\omega_j=\frac{g_j}{\sum_{j=1}^{n}g_j} \tag{7.23}$$

指标的熵值越小，其熵权越小，反映该指标越重要，提供的有用信息量越大。

（6）计算综合代价值 c_i。

$$c_j = \sum_{j=1}^{n} \omega_j p_{ij}, \quad i = 1,2,...,m \tag{7.24}$$

c_i 为第 i 个任务元的综合评价值。

空闲执行器节点收到拍卖执行器节点的任务元信息，计算执行代价后，成为投标执行器节点，返回本节点的执行代价信息。算法描述如下：

```
/*投标策略伪代码*/
/*消息处理函数*/
Msg_handle(msg)
{
    Switch(msg->type){
    case BIDDING:                                   //如果是拍卖消息
        Compute_cost(msg, bidder_inf);              //计算代价
        Send(bidder_inf);                           //返回代价值
        break;
    }
    break;
}

/*计算任务元执行代价*/
Uint16_t Compute_cost(uint16_t* rec_msg, struct Bidder* bidder_inf)
{
    Uint16_t power_re;
    Uint16_t X_coordinate;
    Uint16_t Y_coordinate;

    power_re = Read_res_power();                    //当前剩余能量
    get_location(X_coordinate, Y_coordinate);       //当前地理位置
    for(i=0; i < rec_msg->tu_number; i++)
    {
        bidder_inf->cost[i] = compute_cost(power_re, X_coordinate, Y_coordinate,
                rec_msg->tu_inf[i]);                //计算执行 m 个任务元的代价
    }
    return;
}
```

7.3.3　无工序限制的任务指派

拍卖执行器节点收到各投标执行器节点的代价值以后，进行评估，安排最优执行组合。

假设一个 MAT 任务可分解成 n（n>1）个任务元，由 m（m>1）执行器节点完成，并且每个任务元都由一个执行器节点完成，任务元本身没有工序限制，也就是相互之间没有执行的先后顺序限制，各执行器节点可以独立执行各任务元。招标执行器节点通过各执行器节点发回的标的，合理分派任务，使总代价最小。这是一个典型的分派问题[99]（Assignment Problem，AP）。

设 C 为分派方案的总代价，c_{ij} 为第 i 个执行器节点执行第 j 个任务元所需要的代价，$c_{ij}\to\infty$ 表示不可能胜任，则分派问题可以表示成

$$\min C=\sum_{i=1}^{m}\sum_{j=1}^{n}c_{ij}z_{ij}\ ,（m=n）\tag{7.25}$$

$$\text{ST}\quad \sum_{j=1}^{n}z_{ij}=1,\ \ i=1,2,...,m$$

$$\sum_{i=1}^{n}z_{ij}=1,\ \ j=1,2,...,n$$

式中，z_{ij}=0 或者 1，表示第 i 个执行器节点没有分派或者分派第 j 个子任务。$m=n$ 表示执行器节点数量与子任务数量相同，属于平衡分派问题模型，利用“匈牙利”法解题。

式（7.25）在 m>n 情况下，可转化为平衡分派问题模型：

$$\min C=\sum_{i=1}^{m}\sum_{j=1}^{n+k}c_{ij}z_{ij}\ ,（m=n+k）\tag{7.26}$$

$$\text{ST}\quad \sum_{j=1}^{n}z_{ij}=1,\ \ i=1,2,...,m$$

$$\sum_{i=1}^{n}z_{ij}=1,\ \ j=1,2,...,n+k$$

式（7.25）在 m<n 情况下，可转化为平衡分派问题模型：

$$\min C=\sum_{i=1}^{m+l}\sum_{j=1}^{n}c_{ij}z_{ij}\ ,（m+l=n）\tag{7.27}$$

$$\text{ST}\quad \sum_{j=1}^{n}z_{ij}=1,\ \ i=1,2,...,m,m+1,...,m+l$$

$$\sum_{i=1}^{n}z_{ij}=1,\ \ j=1,2,...,n$$

7.4　实时协作（RC）算法流程

RC 算法针对事件驱动的 AA 协作，由检测出事件的执行器节点充当拍卖执行器节点，根据事件类型进行任务分解，并利用功率控制技术，基于事件发生频率和基本任务元数量进行动态范围内的招标，以便网络能耗均衡；收到招标信息的执行器节点根据自身情况进行基于熵权的代价评估，并将代价值返回拍卖执行器节点；拍卖执行器节点根据各返回值进行任务分派，并发出执行指令。整个流程如图 7.4 所示。

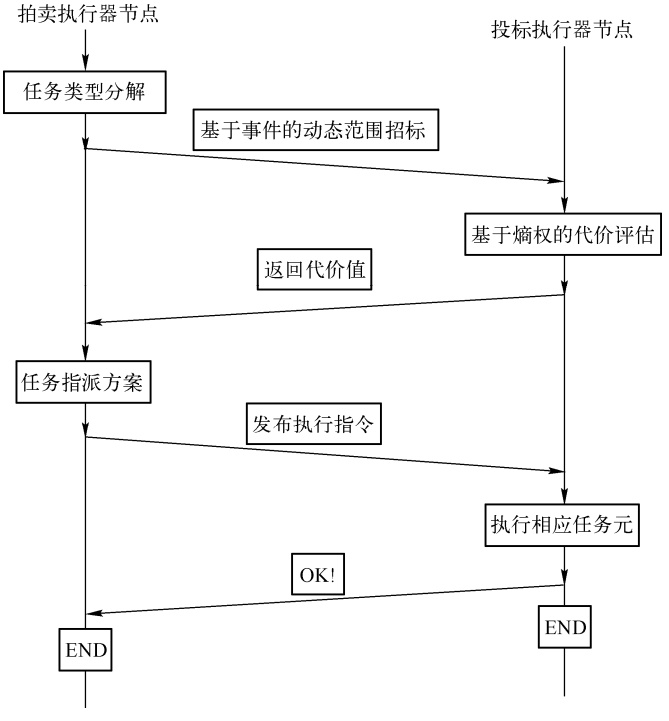


图 7.4　RC 算法流程图

Fig.7.4　RC algorithm flowchart

7.5　算法仿真与性能分析

7.5.1　仿真环境与参数

为验证 RC 算法性能，本书利用 MATLAB 和 NS-2 仿真平台，针对网络通

信开销、任务完成时间、能耗均衡和生存期等指标，与典型文献[5]的 KE 算法和文献[7]中的 DARA 算法开展对比仿真实验。监控区域 200m×200m，节点部署如 3.5.2 节所述，200 个静止传感器节点随机布置，16 个具有移动功能的执行器节点通过虚拟力算法均匀布置。本书假设火灾报警事件，由传感器节点在一定时间内按给定概率随机产生并向本簇执行器节点报告。该事件对应的火灾救援任务分解见表 7.2，其他相应实验参数见表 7.3。

表 7.2　火灾救援任务

Table 7.2　Fire protection task

任务描述	对事件发生地进行洒水动作，并发出声，光报警信号
任务类型	无工序多执行器任务（FE-MAT）
任务元分解	洒水、声音报警、闪光报警三个任务元
任务元能耗	洒水：1J。声音报警：0.5J。闪光报警：0.5J

表 7.3　实验参数

Table 7.3　Test parameters

监 控 区 域	200m×200m	信 道 模 型	TwoRayGround
无线标准	802.11	执行器节点数量	16
传感器节点数量	200	簇半径	25m
执行器节点工作半径	5m	执行器节点移动能耗	0.01J/m
执行器节点初始能耗	300J	事件发生频率	5 次/min
执行器节点移动速率	10m/s	任务元执行时间	0.5s

执行器节点的动态招标范围由发送功率控制，根据 TwoRayGround 模型中发送功率计算公式：

$$P_{\mathrm{t}}=\frac{P_{\mathrm{r}}d^4L}{G_{\mathrm{t}}G_{\mathrm{r}}h_{\mathrm{t}}^2h_{\mathrm{r}}^2} \tag{7.28}$$

NS-2 仿真器中网络物理接口的默认的接收门限：

Phy/WirelessPhy set RxThreash_=3.65262e-10，以及天线长度、高度和发送接收增益，接收功率与距离的关系见表 7.4。

表 7.4　接收功率与距离

Table 7.4　Rxpower VS distance

距离（m）	接收功率（W）
15	2.81838e-5
20	8.91754e-6

续表

距离（m）	接收功率（W）
22	6.0908e-6
30	1.7615e-6
60	1.1009e-7
120	6.8808e-9

根据表 7.4 以及式（7.12），可以算出不同传输距离下的发送功率（详细细节可参考 tworayground.cc 文件）。

7.5.2　算法通信开销

RC 算法通信开销主要集中在两个部分。在招标阶段：拍卖执行器节点以一定功率发出任务信息，第 i 次各投标执行器节点 Ni 算出评价值并返回，所以第 i 次招标的通信开销为 $C_{i\text{-bidding}}=1+Ni$，如果招标次数为 m，则招标阶段总通信次数：

$$C_{\text{bidding}} = \sum_{i=1}^{m}(1+Ni)$$

通信开销分三种情况考查如图 7.5 所示。

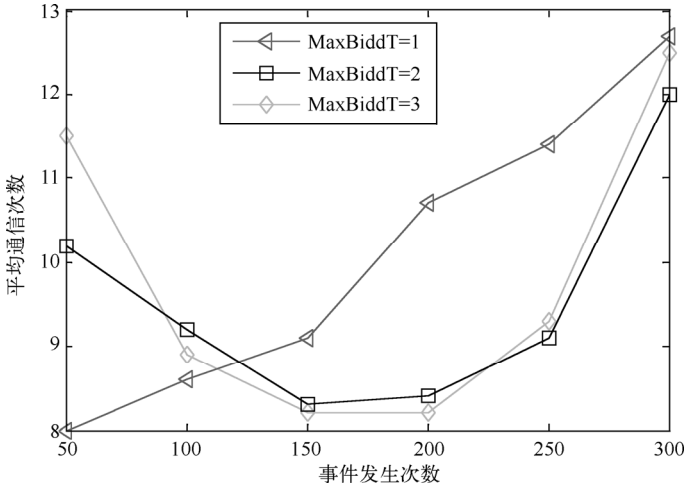


图 7.5　算法通信开销对比

Fig. 7.5　Communication overhead comparison among different Max_Bidding_Times

MaxBiddT 表示允许最大失败次数：当 MaxBiddT=1 时，算法无论首次招标成功或者失败，通信次数主要都由增加的候选节点数决定，呈线性增长；当

MaxBiddT=2 时，首次招标次数扩大，导致初期通信开销相对变大，但是由于获得了较为精确的先验数据，在以后的招标中，首次招标成功的几率变大，相应的通信次数减少，到后期由于各节点剩余能量减少，一次招标成功几率变小，通信次数增加较快；MaxBiddT=3 的总体趋势与 MaxBiddT=2 类似，由于允许失败次数更大，初期的通信次数也更多，获得的先验数据也更加精确。

7.5.3 任务完成时间

为检验 RC 算法的实时性，将 MaxBiddT 设置为 1，并与 KE 算法和 DARA 算法进行比较，如图 7.6 所示。

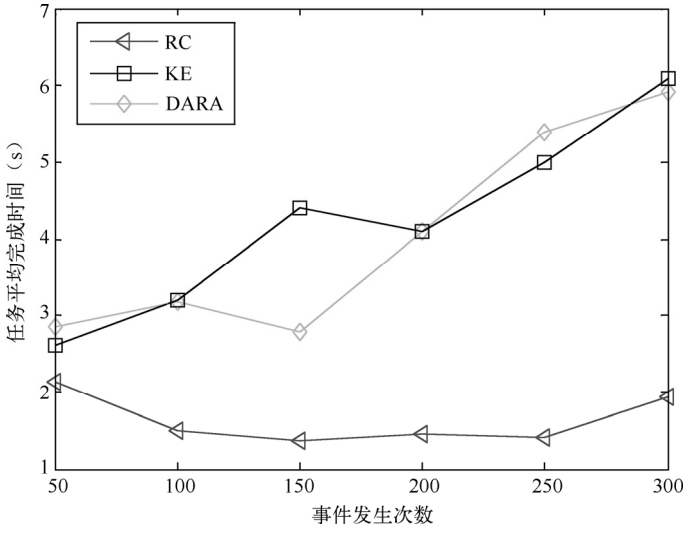


图 7.6 任务平均完成时间比较图
Fig.7.6 Comparison of average task finishing time

RC 算法在首次招标过程中，通信方面的时延开销较 KE 算法和 DARA 算法大，但是由于任务分解后，各任务元可以并行执行。在任务执行时间远大于通信时延的情况下，RC 算法仍然显示出较好的实时性，任务平均完成成功时间小于其他两种算法。经过招标，在获得了先验数据的情况下，招标成功几率增加，通信时延减少，平均任务完成时间也相应减少。KE 算法和 DARA 算法没有任务分解和任务元并发执行过程，每个任务都由单个执行器节点完成，整体任务完成时间延长，随着事件发生增加，能量消耗出现不均衡，必须重建通信链路以保证任务执行，时延增大。由此显示，RC 算法采取的多执行器节点协作技术，虽然在通信时延上有一定开销，却使得任务并发执行，缩短了任务

整体完成时间。

7.5.4　能耗均衡

为了比较算法能耗均衡问题，引入执行器节点能耗均衡指数，将执行器节点的剩余能量与初始能量比值作为能耗均衡指数：

$$EB_i = E_{i\text{-rest}} / E_{i\text{-init}} \qquad (1 \leqslant i \leqslant N)$$

式中，N 为执行器节点总数。随着监测时间的推移，考察网络中执行器节点其对应的方差 V，如图 7.7 所示。

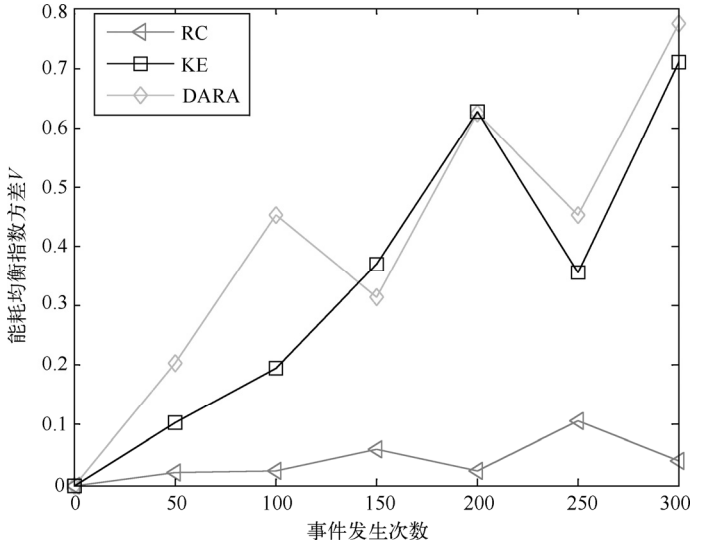


图 7.7　能耗均衡指数方差比较

Fig.7.7　Comparison of energy-balancing index square variance

RC 算法将任务分解后，多执行器节点并发执行，最大限度将执行能耗分摊到各节点上，使得节点整体能耗比较均衡。而对于 KE 和 DARA 算法来说，由于事件随机发生，导致某些执行器节点能耗过快，其均衡指数方差波动较 RC 算法大。

7.5.5　网络寿命

本次仿真将网络寿命定义为从事件发生开始到出现第一个执行器节点能量耗尽的时间，如图 7.8 所示。在 KE 算法和 DARA 算法中，每次事件发生后，监控区域执行器节点必须独立完成相应任务，当发生地点随机出现时，一些区域内事件发生频率可能高于其他区域，导致该区域内的执行器节点执行能耗过

大，网络寿命变短。而 RC 算法通过协作机制，根据网络能耗情况动态调整协作范围，尽量保持能耗均衡，网络寿命明显提高，比 KE 算法增加 38%，比 DARA 算法增加 25.6%。

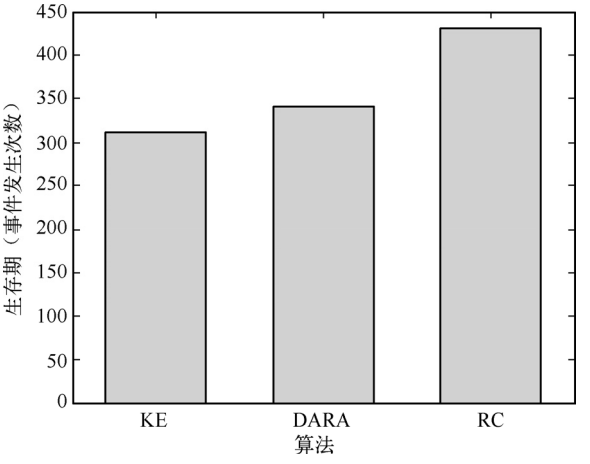


图 7.8　网络寿命比较
Fig.7.8　Comparison of network lifetime

7.6　小结

无线传感器/执行器网络中的 AA 协作主要用于解决任务如何在多执行器节点中的分配问题。典型协作算法或者只关注实时性，而忽略网络能耗导致生存期较短，或者关注网络连通度的快速恢复。本书基于分簇结构，提出一种实时协作算法。通过将任务分解成若干任务单元，利用功率控制技术进行动态范围招标，建立基于熵权的评价标准，根据网络的实际情况分派任务元，使得任务可以并发执行，缩短执行时间，同时尽量将执行能耗分摊到邻居执行器节点，解决了事件频发区域内节点能耗过快问题，延长了网络生存期。

参 考 文 献

[1] Akyldiz F. Kasimoglu I. Wireless sensor and actuator networks: Research challenges[J]. Ad Hoc Networks Journal Elsevier. 2004,2(4):351-367.

[2] Tommaso Melodia Dario Pompili Vehbi C. A Distributed Coordination Framework for Wireless Sensor and Actor Networks. International Symposium on Mobile Ad Hoc Networking & Computing[C]. Proceedings of the 6th ACM international symposium on

Mobile ad hoc networking and computing. 2005:99-110.

[3] Shah G. A. Bozyigit M. Akan O. Real Time Coordination and Routing in Wireless Sensor and Actuator Networks[C]. In Proceedings of the 6th International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking NEW2AN. 2006:365-383.

[4] Ngai C. H. Edith Lyu R. Michael Liu. A Real Time Communication Framework forWireless Sensor-Actor Networks[C]. IEEE Aerospace Conference. 2006. 2021-2029.

[5] M Lopez-Nores J. Garcia-Duque JJ Pazos. Qualitative assessment of approaches to coordinate activities of mobile hosts in ad hoc networks [J]. IEEE Communications Magazine. 2008. 12. 108-111.

[6] K Akkaya F. Senel. Detecting and connecting disjoint sub-networks in wireless sensor and actor networks [J]. Ad Hoc Networks, Elsevier. 2009, 46(7):1330-1346.

[7] Ameer Ahmed Abbasi Mohamed Younis. Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks [J]. IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS. 2009. 20(9): 1366-1379.

第 8 章　基于 AA 协作的单目标任务分派算法

传统的无线传感器网络通过部署大量的传感器节点监测环境参数，功能比较单一。随着功能更强大的执行器节点被整合到网络中，不仅可以对多种传感器数据进行监测、分析处理，同时还能根据事件类型做出的复杂反应。无线传感器/执行器网络的基本功能不但涵盖了控制、监督及跟踪等环境型应用，未来更会出现在满足商业市场的需求上，例如火警监测、大楼照明自动化、楼宇空调控制、交通流量控制管理等。如何在这些应用中，做出更快的反应、执行更复杂的逻辑关联性的任务，将成为未来在无线传感器/执行器网络中的一大议题。典型应用是数据采集与监视控制系统（Supervisory Control And Data Acquisition，SCADA）[1]。SCADA 系统的应用领域很广，可以应用于电力系统、给水系统、石油、化工等领域的数据采集与监视控制以及过程控制等诸多领域。如图 8.1 所示，对于城市下水道管网排水（Combined Sewer Overflow, CSO）问题，管网规模大，覆盖面广，情况复杂。由于设计或者其他原因，一旦发生暴雨，常容易引起路段积水，严重者甚至威胁生命和财产安全。利用部署大量传感器进行检测，发生溢流事件后，通过无线网络报告给汇聚节点，安排相应管道执行器（排水阀）及时排水，杜绝积水问题发生。

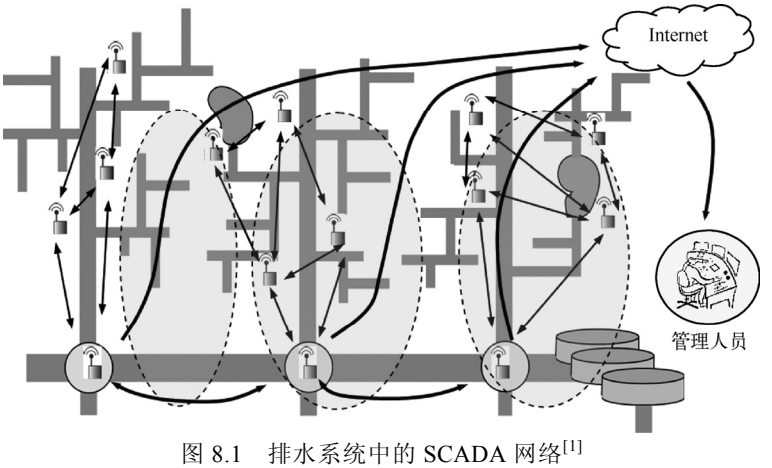


图 8.1　排水系统中的 SCADA 网络[1]

Fig.8.1　SCADA of drainage system[1]

典型的下水管执行器任务由多个执行器（排水阀）打开（关闭）操作组成，各个执行器阀门开启量也不相同，且所有操作需要符合一定顺序，以保证水流顺利排出，如图 8.2 所示。

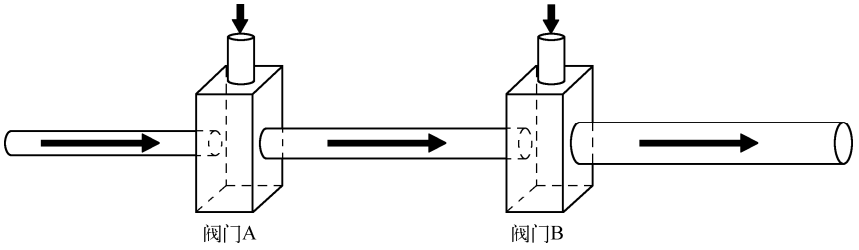


图 8.2　典型下水管执行器任务

Fig.8.2　Typical task of manhole

AA 协作中主要解决任务分派问题，但是对于多个有工序限制（ordered execution）任务在多执行器节点上的分派执行，目前协作算法在这方面很少研究。本书针对任务在各执行器节点的协作问题，提出一种面向 AA 协作的单目标任务调度策略（Single Objective Task Scheduling，SOTS）。根据执行器节点的剩余能量和工作状态，在任务时效期内，安排各任务在执行器节点上的执行周期，通过构建最小化最大完成时间目标函数，并以节点剩余能量作为约束条件，利用微粒群（Particle Swarm Optimization，PSO）优化算法，搜索最优任务调度方案。

8.1　单目标任务分派

8.1.1　最小化最大任务完成时间

这里，本书将完成一个特定目标的行为定义为一个任务，包括通信、数据处理、执行动作等行为。

【定义 8.1】 一个由单个执行器节点完成的任务，称为单执行器任务（Single Actor Task，SAT）；一个任务由多个执行器节点共同完成并具有任务约束条件，则称为多执行器任务（Multi Actor Task，MAT）。

【定义 8.2】完成一个特定目标的操作（包括通信、数据处理、执行动作等），执行时间和能耗可预知，称为一个任务执行单元，简称任务元，记为 t。

【定义 8.3】一个多执行器任务可分为多个任务执行单元，每个任务元表示在一个执行器节点上的一次执行，一个任务 $T = t_1 \bigcup t_2 \bigcup \ldots \bigcup t_n$，且 $\forall t_i, t_j \in T$，$\exists t_i \bigcap t_j = \phi$，称任务 T 具有可分性。

【定义 8.4】若 $O=\{o_1,o_2,\ldots,o_n\}$ 为任务 T 的 n 个任务元执行顺序，o_1 只有一个后继任务元，o_n 只有一个前趋任务元，其余任务元拥有一个前趋和一个后继，称其执行具有工序限制（Ordered Execution，OE）。

考虑 n 个任务在 m 个执行器节点上的执行过程，每个任务在各执行器节点上的完成时间已知，并且每个任务必须按照事先规定的顺序完成其在各执行器节点上执行，即任务约束条件。如果一个任务只需一个执行器节点完成，则该任务在其他执行器节点的执行时间用零表示。要求确定在每个执行器节点上执行的所有任务的顺序或开工与完工时间，使得在符合各任务的约束条件的前提下最优化最大完成时间。这里假设：一个执行器节点同时只能执行一个操作；任务没有抢先执行的特权；每一项操作的执行时间和能耗可以估算；操作结果的数据报文格式一致，即传输结果的通信能耗一致。

设 $p_{i,j}$ 为任务 i 在执行器节点 j 上的执行时间，$T(j_i,k)$ 为任务 j 的第 i 个任务元在执行器节点 k 的执行完成时间，$\pi=(j_1,j_2,\ldots,j_n)$ 为所有任务的一个排序，$\prod$ 为所有排序的集合。

一般地，无线传感器/执行器网络中一个任务元的执行时间 t_{oper} 包括：指令下达时间 t_{comm}、数据采集/执行动作时间 t_{colle}、数据处理时间 t_{proc}。

$$p_{i,j}=t_{\text{oper}}=t_{\text{comm}}+t_{\text{colle}}+t_{\text{proc}} \tag{8.1}$$

则各任务在每个执行器节点上完成时间描述如下：

$$T(j_1,1)=p_{j_1,1} \tag{8.2}$$

$$T(j_i,1)=T(j_{i-1},1)+p_{j_i,1}x_{j_i,1},i=2,\ldots,n \tag{8.3}$$

$$T(j_i,k)=T(j_1,k-1)+p_{j_1,k}x_{j_1,k},k=2,\ldots,m \tag{8.4}$$

式中，$x_{j_i,k}=1$ 表示该执行器节点 k 被选中执行 j_i 任务元；$x_{ji,k}=0$ 表示没被选中。式（8.2）表示执行器节点 1 执行任务 j 第一个任务元的完成时间等于执行时间。式（8.3）表示执行器节点 1 执行 j_i 任务元的完成时间等于执行 j_{i-1} 任务元的完成时间加上 j_i 任务元的执行时间。式（8.4）同理。

$$T(j_i,k)=\max\{T(j_{i-1},k),T(j_i,k-1)\}+p_{j_1,k}x_{j_1,k},\ i=2,\ldots,n;\ k=2,\ldots,m \tag{8.5}$$

式（8.5）表示执行器节点 k 执行任务元 j_i 必须在执行器节点 k 已经执行完上一个任务元 j_{i-1} 之后，同时任务元 j_i 在上一个执行器节点 k–1 的任务元也已经完成的基础上进行。

则最大完成时间：

$$T_{\max}(\pi)=T(j_n,m) \tag{8.6}$$

最小化最大完成时间的调度方案：

$$\min f=\min\{T\max(\pi)\}\quad \forall\pi\in\Pi \tag{8.7}$$

8.1.2　执行器节点剩余能量约束

设 $P(t)$ 为单位时间内执行器节点执行操作所需能耗，γl 表示采集 l 长度数据所需能耗，则执行器节点 k 所需执行能耗：

$$E_{k\text{-action}}=\sum_{j=1}^{n}\sum_{i=1}^{m}(P(t)P_{j_i,k}+\gamma l)x_{j_i,k} \tag{8.8}$$

可行分配方案必须考虑到执行器节点剩余能量的约束，每个执行器节点的剩余能量不但要足以完成所有操作序列，还要将执行结果数据传给下一执行器节点或汇聚节点。根据发射硬件能耗模型[2]，r 为执行器节点通信距离，l 表示帧长，则传输和接收能耗公式为

$$\begin{cases}E_{tx}(r,l)=(ar^{n'}+\beta)l\\ E_{rx}(l)\beta l\end{cases} \tag{8.9}$$

式中，$ar^{n'}$ 取决于在距离 r 传输的发射功率；β 是发射电路（如 PLLs、VCOs）能耗；n' 为信道衰减倍数，取决于环境；则一个执行器节点 k 所有操作所需的总能耗：

$$E_{k\text{-all}}=\sum_{j=1}^{n}\sum_{i=1}^{m}(P(t)p_{j_i,k}+\gamma l+(ar^{n'}+\beta)l+\beta l)x_{j_i,k} \tag{8.10}$$

执行器节点 k 的能量约束条件为需要承担任务的总能耗必须小于等于剩余能量：

$$E_{k\text{-all}}\leqslant E_{k\text{-rest}}\ ,\ k=1,2,\ldots,m \tag{8.11}$$

8.2　SOTS 算法

解决上述带能量约束条件的最小化最大完成时间问题，本书采用微粒群优化算法，利用该算法全局搜索能力强的特点，与 NEH（Nawaz-Enscore-Ham）算法的局部改良能力相结合，提高微粒群算法的搜索精度，寻找全局最优解。由于任务分配问题需要映射到微粒的位置矢量上，所以首先需要进行编码，同时利用能量约束条件确定执行器节点的角色，并借助 NEH 方法增强局部搜索能力，得到理想的任务分派方案。

8.2.1 执行器节点角色确定

对于 n 个有工序限制任务在 m 个执行器节点上执行问题，调度解的数量巨大。

【命题 8.1】 n 个任务在 m 个执行器节点上的执行问题，包含 $A^n_{n\times m}$ 种排列。

【证明】 设一个任务最多可以分解成 m 个执行单元，则 n 个任务最多可分解的任务元总数为 $n\times m$，在 $n\times m$ 个任务元中选出 n 个执行，其排列数为 $A^n_{n\times m}$，得证。

为了降低计算的复杂度，算法在任务分派前，首先发布任务情况，根据各执行器节点返回的情况，确定各自承担的角色，在一定时间内固定执行某一类型的任务元，以便降低排列数。

【命题 8.2】 执行器节点角色固定后，n 个任务在 m 个执行器节点上的执行问题，其排列种数降为 $n!$。

【证明】 当执行器节点角色固定，每个执行器节点着重关心 n 个任务元的执行顺序，其全排列数为 $n!$，得证。

执行器节点的角色只在一定时间内充当，时间长了会造成能耗不均衡，周期性发布节点信息，重新确定角色是必须的。同时，对于角色的确定，必须符合公式（8.11）的能量约束，节点的剩余能量必须满足执行 n 个某种类型任务元的能耗要求。否则，必须重新确定角色。很显然，此时的排列数仍然比较大，不利于满足无线传感器/执行器网络的实时性要求。约定每个执行器节点上所有任务的执行顺序相同，则排列数降为 $n!$。研究在 $n!$ 种排列中，选出一种最佳执行顺序，使得全网的最大完成时间最小，仍然具有重要的理论意义和工程价值。

8.2.2 标准微粒群优化算法

微粒群优化算法（Particle Swarm Optimization，PSO）[3]是一种进化计算技术，由 Eberhart 博士和 kennedy 博士发明。与其他进化类算法类似，PSO 也采用“群体”与“进化”的概念，同样也是依据个体（粒子）的适应值大小进行操作。不同之处在于 PSO 算法并不对个体使用进化算子，而是将每个个体看成在 d 维搜索空间中的一个没有质量和体积的微粒，并在搜索空间中以一定的速度飞行。基本原理是：PSO 首先在可行解空间和速度空间随机确定微粒的初始位置和初始速度，其中位置用于表征问题解。

$X_i=[x_{i,1},x_{i,2},...,x_{i,d}]$ 为 d 维搜索空间的第 i 个微粒的位置；

$V_i=[v_{i,1},v_{i,2},...,v_{i,d}]$ 为 d 维搜索空间的第 i 个微粒的速度；

$P_i=[p_{i,1},p_{i,2},...,p_{i,d}]$ 为 t 时刻 d 维搜索空间的第 i 个微粒的最佳位置，由微粒的适应性函数 $finess(X)$ 决定，函数值越小代表微粒适应性越好。微粒 i 的最佳位置由式（8.12）确定：

$$p_i(t)=\begin{cases}p_i(t-1) & \text{如果} \quad \text{fitness}(x_i(t))\geqslant\text{fitness}(p_i(t-1))\\ x_i(t) & \text{如果} \quad \text{fitness}(x_i(t))<\text{fitness}(p_i(t-1))\end{cases} \tag{8.12}$$

式中，t 表示迭代次数。再根据每个微粒所经过的最佳位置（P_{best}）确定群体最佳位置 $p_{\text{g}}(t)$，并按如下公式分别更新各微粒的速度和位置：

$$v_{i,j}(t+1)=wv_{i,j}(t)+c_1r_1[p_{i,j}(t)-x_{i,j}(t)]+c_2r_2[p_{\text{g},j}(t)-x_{i,j}(t)] \tag{8.13}$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1),\ j=1,...,d \tag{8.14}$$

式中，w 为惯性权因子;c_1 和 c_2 为正的加速常数;r_1 和 r_2 为在 0 到 1 之间均匀分布的随机数。

8.2.3　基于 ROV 规则的编码

由于 PSO 算法中微粒的位置为连续值矢量，本身无法表示任务的执行排序，因此必须构造从微粒位置矢量到任务排序的恰当映射，是利用 PSO 算法解决任务分派的首要问题。

微粒的位置矢量 $\boldsymbol{X}_i=[x_{i,1},x_{i,2},\ldots,x_{i,n}]$ 可以由随机键编码[4]产生，即在一定范围内随机产生一些实数。但是这些用实数表示的位置分量本身无法表示任务的执行顺序，因此必须进行转换。为了保证编码策略不遗漏问题的全局最优解，并适应标准 PSO 算法。引入一种针对随机键编码的基于升序排列（Ranked Order Value，ROV）规则，实现从微粒的连续位置矢量到任务排序的转化，使之适应问题求解。基于随机键编码产生的位置分量虽然是一些实数，但是分量之间存在大小差异，利用这个差异可将连续的位置矢量转化为离散的执行排序，具体规则如下：

（1）找出 $x_{i,1},x_{i,2},\ldots,x_{i,n}$ 中值最小的分量 $x_{i,d}$，并令对应的加工排序值 j_d 的 ROV 值为 1。

（2）找出除开 $x_{i,d}$ 的值最小的分量，并令其对应的 ROV 值为 2；重复直到找完所有位置分量。

此时，微粒的位置矢量 $\boldsymbol{X}_i=[x_{i,1},x_{i,2},\ldots,x_{i,n}]$ 转换为离散的执行排序 $\pi=(j_1,j_2,\ldots,j_n)$。需要注意两点：第一，如果位置矢量中出现多个相同值的分量，需要随着位置的增加依次累加一个足够小的正数，使得这些位置的分量值互不相同；第二，对于局部搜索操作，如互换操作 SWAP、逆序操作 INVERSE 和插入操作 INSERT[5]，位置分量与对应的 ROV 值是一一对应的，排序发生变化，位置分量相应的也发生变化，调整非常简单。

8.2.4　基于 NEH 方法的局部搜索

研究表明，进化算法包括 PSO 算法的全局粗搜索能力胜于其局部细搜索能力，

因此，SOTS 算法首先利用标准 PSO 算法的进化操作进行对解空间的群体粗搜索，即利用式（8.13）、式（5.14）在连续空间上更新各微粒的速度和位置信息。然后利用 ROV 规则将连续空间下的各微粒位置转化成对应的任务排序，并利用适应性目标函数 5.7 进行评价，得到每个微粒的局部最佳位置 P_{best}。在从这些微粒中按照一定概率选出若干实施基于 NEH 方法的邻域局部搜索，以期获得更好的解，并更新局部最佳位置 P_{best} 和群体全局最佳位置 G_{best}。

NEH 方法是一种有效的构造型算法，其过程是：首先计算各任务在所有执行器节点上的执行时间总和，并按照递减顺序排列，先将前两个任务进行最优调度，然后依次将剩余任务逐一插入到已调度好的任务排列中的某个位置，使得子调度的总完成时间最小，直到所有任务均调度完毕，最后得到一个调度方案。

/*NEH 方法的局部任务调度*/

步骤 1　给定所有任务的一个排序 π 。

步骤 2　令 $k=1$，取出 π 中的前 2 个任务进行调度，确定其执行任务的最大完成时间最小的序列，并作为当前序列。

步骤 3　令 $k=k+1$，取出 π 中接下来的一个任务，分别插入当前序列中的各个可能位置，最终确定在各执行器节点上执行这些任务的最大完成时间最小所对应的序列，并作为当前序列。

步骤 4　重复步骤 3，直到 π 中所有任务均得到新的排序操作。

局部搜索显然有助于提高找出最优解的质量，但同时也占用了一定的计算时间和资源，为了平衡算法的全局搜索和局部搜索能力，本书设计一种基于平衡的概率选择策略，首先计算所有微粒最佳位置 P_{best} 的选择概率，再根据此按照轮盘赌选择机制进行多轮选择。对于被选中的 P_{best}，进行基于 NEH 方法的局部搜索。

/*基于平衡的概率选择策略*/

步骤 1　将所有微粒按其 P_{best} 对应的任务排序目标函数递减顺序确定序号 R_i，目标值最大的微粒序号为 1。

步骤 2　确定所有 R_i 号微粒对应的概率 $\text{prob}_{R_i}=c(1-c)^{R_i-1}$，其中 c 是序号为 1 的微粒概率（c 可以自行确定）。

步骤 3　计算所有微粒的选 s 择概率 $\sum_{j=1}^{R_s}\text{prob}_{R_i}$ 。

步骤 4　根据各微粒 P_{best} 的选择概率，按照轮盘赌选择机制，生成 0 到 1 之间的随机数与每个微粒的选择概率进行匹配，被选中的 P_{best} 进行基于 NEH 方法的局部搜索。

显然，上述方法中，好的个体最佳位置对应的排列得到局部改进的概率更大，而群体的最佳位置 P_{best} 对应的排列获得局部搜索的概率最大。为调整局部搜索的计算开销，还可以在基于平衡的概率选择策略中的步骤 4，增加对被选中的 P_{best}，

按照一定概率进行局部搜索加以控制，而不是一定进行局部搜索。

8.2.5　算法流程和分析

根据上述设计，面向 AA 协作的微粒群任务分派算法模块示意图如图 8.3 所示。

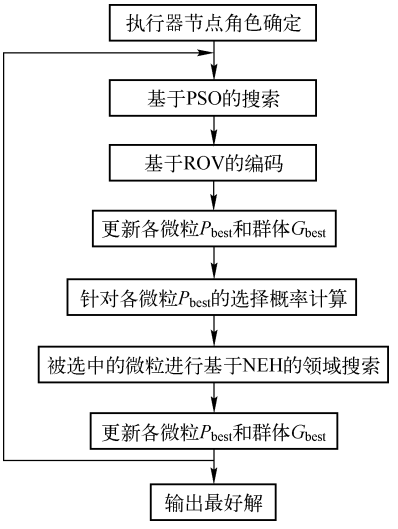


图 8.3　基于 AA 协作的单目标任务分派算法示意图

Fig.8.3　SOTS algorithm

下面给出算法的具体步骤：

/*基于 AA 协作的单目标任务分派算法伪代码*/

1．random initialize $x_i(0)$，$1 \leqslant i \leqslant p_s$　// p_s 为群体规模

2．get task sequence for each partical by ROV coding

3．compute $fitness(x_i(0))$ for each partical from equation 5.6

4．random initialize $v_i(0)$

5．set $P_{best_i} = x_i(0)$, and $G_{best} = \min(fitness(x_i(0)))$

6．$noChangeGen = 0$

7. while $noChangeGen < L$ do　//重复执行下列步骤，直到群体的 *gbest* 目标值连续 L 步保持不变

8．update $x_i(t)$, $v_i(t)$ from equation 5.13 and 5.14

9．get task sequence for each partical by ROV coding

10．compute $fitness(x_i(t))$ for each partical from equation 5.6

11．update P_{best_i} and G_{best}

12．compute Selection probability for P_{best_i}

13．run NEH neighborhood search for selected partical P_{best_j} and $NEH(P_{best_i})$

14．update G_{best}

15．if $gbest \neq NEH(pbest_i)$, $noChangeGen = 0$

16．else $noChangeGen = noChangeGen + 1$

17．output G_{best} and corresponding task sequence and $fitness(G_{best})$

【定义 8.5】在任务调度的搜索空间 Ω 内，G_{best} 为算法在第 t 次进化中求得的最佳位置，G_{best}^{*} 为 Ω 中的固定位置，若有

$$\lim_{t\to\infty} G_{\text{best}}(t) = G_{\text{best}}^{*} \tag{8.15}$$

说明求解过程达到一个稳定状态，称为该算法收敛。如果 G_{best} 为全局最佳位置，则算法获得了全局最优解；否则，算法陷入局部最优。

【定理 8.1】基于 AA 协作的单目标任务分派算法是收敛的。

【证明】假设搜索空间是有限的，群体的每个微粒的当前位置为一个状态，群体下一代所处状态的概率仅取决于当前状态，而与以前状态无关。因此，算法的搜索过程可以看作一个群体规模为 k 的有限状态 $(S=\{s_1,s_2,\ldots,s_k\})$ 的马尔可夫链。

当问题规模为 m，最佳调度为 s^{*}，所有群体状态构成一个 $1\times N$ 的向量，其中 $N=(k\times m)!$，由此可以得到群体的状态向量 $\boldsymbol{M}$：

$$\boldsymbol{M}=\begin{vmatrix} s_1^1 & s_2^1 & \cdots & s_k^1 \\ s_1^2 & s_2^2 & \cdots & s_k^2 \\ \cdots & \cdots & \cdots & \cdots \\ s_1^N & s_2^N & \cdots & s_k^N \end{vmatrix} \tag{8.16}$$

假设经过若干代进化后，已达到群体最优状态 s^{*}，则可以表示为

$$\boldsymbol{M}'=\begin{vmatrix} s_1^{*} & s_2^{*} & \cdots & s_k^{*} \\ s_1^2 & s_2^2 & \cdots & s_k^2 \\ \cdots & \cdots & \cdots & \cdots \\ s_1^N & s_2^N & \cdots & s_k^N \end{vmatrix} \tag{8.17}$$

由式（8.13）和式（8.14）可知，当群体处于最优位置时，下一次进化中求得的微粒当前位置不变，状态转移矩阵可写为

$$\boldsymbol{M}=\begin{vmatrix} 1 & \boldsymbol{M}^{N-1} \\ \boldsymbol{R} & \boldsymbol{T} \end{vmatrix} \tag{8.18}$$

式中，$\boldsymbol{M}$ 为随机矩阵，每个行向量至少含有一个正元素。$\boldsymbol{M}^{N-1}$ 为 $N-1$ 维向量，显然 $\boldsymbol{M}$ 为可约随机矩阵，且 $\boldsymbol{R}$ 和 $\boldsymbol{T}$ 为非零矩阵，根据可约随机矩阵的性质有 $M^{\infty}=(M_1^{\infty},0,\ldots,0)$，其中 $M_1^{\infty}>0$，则状态分布 $M_1^{\infty}=1$，于是 $M^{\infty}=(1,0,\ldots,0)$，表明每个状态以概率收敛于全局最优状态 s^{*}，得证。

【定理 8.2】基于 AA 协作的单目标任务分派算法的空间复杂度为 $\mathrm{O}((5p_s+2)n)$，群体每次进化的时间复杂度为 $\mathrm{O}(mn+p_s+5Lp_sn)$，其中 p_s 为群体规模，n 表示任务数量，m 为执行器节点数量，L 为进化次数。

【证明】在基于 AA 协作的单目标任务分派算法过程中，每次进化需要存储

p_s 个微粒的当前位置、微粒最佳位置、速度和适应目标函数值，所消耗的存储空间为 $O(4p_sn)$；NEH 局部搜索时需要计算每个微粒的选择概率，存储空间为 $O(p_sn)$；邻域搜索时还需要一个存储新位置或速度的空间，加上一个存储群体最优调度，合为 $O(2n)$。所以算法的空间复杂度为 $O((5p_s+2)n)$。

执行器节点角色确定，时间复杂度为 $O(mn)$，初始化 p_s 个微粒的时间复杂度为 $O(p_s)$，每个微粒确定历史最优微粒和全局最优微粒、NEH 局部插入操作、ROV 编码、计算选择概率和计算适应函数的时间复杂度均为 $O(p_sn)$，则一次进化的时间复杂度为 $O(4p_sn)$，进化过程要执行 L 次，则重复进化过程中的时间复杂度为 $O(5Lp_sn)$，算法的总时间复杂度为 $O(mn+p_s+5Lp_sn)$。

8.3　算法仿真与性能分析

8.3.1　实验参数

为了考察算法解决执行器节点任务分派的性能，本节利用 9 个不同规模的标准 PFSP 问题，并加以改造，增加能量消耗条件，即每个单位时间消耗的能量为 5000nJ，同时规定单位时间为毫秒（ms），使之更符合在无线传感器/执行器网络环境下的性能考察。9 个测试问题 Rec01～Rec17 从 Reeves[6]设计的一系列算例中选出，定义了任务在各执行器节点的完成时间和执行顺序，以检验算法的性能和网络性能及执行器节点能耗分布情况。

仿真硬件环境：Intel Pentium 4/2.2GHz，512MB RAM。

仿真软件平台：Windows XP，MATLAB 7.0。

算法参数设置见表 8.1。

表 8.1　算法实验参数
Table8.1　Parameters in algorithm

群体规模 P_s	20	惯性权因子 w	1.0
加速因子 c_1	2	加速因子 c_2	2
微粒最小位置值 $x_{\min}$	0	微粒最大位置值 $x_{\max}$	4.0
微粒最小速度值 $v_{\min}$	−4.0	微粒最大速度值 $v_{\max}$	4.0
NEH 局部搜索概率 P_{ls}	0.06	终止条件参数 L	10

8.3.2　算法性能实验

算法性能实验主要考察算法本身的收敛性和有效性，每个算例各独立运行 15 次，表 8.2 列出了算法在各算例中的运算结果和运行时间（s）。

表 8.2　算法性能统计表
Table 8.2　Algorithm performance

问　题	n，m	已知最好解	SOTS 算法			
			最　好　解	平　均　值	最　差　解	平均 CPU 时间（s）
Rec01	20，5	1247	1247	1254	1273	1.93
Rec03	20，5	1109	1109	1114	1142	1.83
Rec05	20，5	1242	1242	1263	1287	1.75
Rec07	20，10	1566	1566	1585	1643	2.05
Rec09	20，10	1537	1537	1573	1631	3.42
Rec11	20，10	1431	1431	1474	1526	3.16
Rec13	20，15	1930	1930	2052	2106	3.87
Rec15	20，15	1950	1950	2037	2076	4.11
Rec17	20，15	1902	1902	1951	1989	3.89

由表 8.2 可见，所有算例都能得到最优解，重复实验的平均值也非常接近已知最优解，表明算法具有很好的搜索性能。

图 8.4 则显示出不同算例，由于执行顺序安排不同，参与执行器节点数量不同，复杂度也各不相同，导致进化代数也不一样。Rec05 算例收敛最快，平均第 26 代即获得最优解。而算例 Rec15 的执行器节点数量增加三倍，使得解空间加大，增加了搜索的难度，进化代数也延长至 62 代。说明影响收敛的两个主要因素：其一是参与执行器节点的数量，它直接导致求解的组合数变化；其二是执行顺序的安排，在任务数量和参与执行器节点数量一定的情况下，不同的执行顺序会导致解空间的变化，也会影响搜索速度。

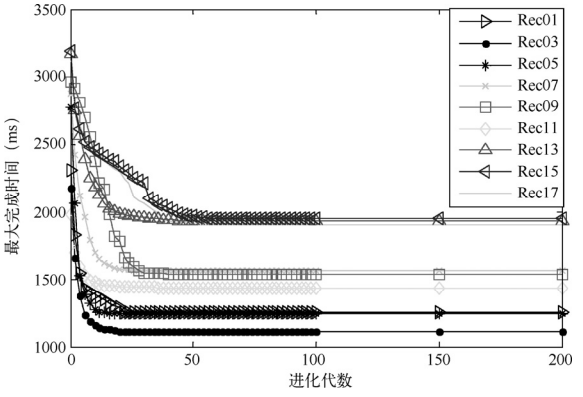


图 8.4　算法收敛图
Fig.8.4　Algorithm convergence

8.3.3　网络性能实验

网络性能实验主要将本算法与典型算法 RT-Maps、EBTA 进行比较，考察响应时间和能耗均衡两方面的情况，实验安排 10 个执行器节点，随机部署在 100m×100m

的范围内。为集中考察算法在任务分派上的性能，排除通信中断引起的多余能耗，每个节点的通信半径都设为 100m，遵循无线通信协议 IEEE 802.11。同时对算例中的单位时间统一成毫秒（ms），便于统计任务最终完成时间。其他参数设置见表 8.3。

表 8.3　执行器节点参数表

Table8.5　Parameters of actor

执行器空闲等待能耗	0J	通信距离 r	100m
数据包长 l	128Byte	发送/接收电路能耗 β	$50nJ\cdot bit^{-1}$
放大倍数 α	$0.0013pJ\cdot(bit\cdot m^{-2})^{-1}$	Two-ray ground 模型指数 n	4
采集能耗指数 γ	$5nJ\cdot bit^{-1}$	单位时间内执行能耗 $P(t)$	5000nJ

10 个执行器节点的初始能量见表 8.4。

表 8.4　执行器节点初始能量

Table8.4　Initial energy of actor

执行器号/初始能量（J）	1/1	2/3	3/1.5	4/2.5	5/2	6/1	7/2	8/2.5	9/2	10/1

任务完成时间包括任务分派的计算时间、通信时间和任务执行时间。从图 8.5 观察，由于本算法主要以最大完成时间为优化目标，通过合理安排任务执行顺序，达到各任务在各执行器节点之间并发执行的目的，任务执行时间明显少于其他两种算法。RT-Maps 算法和 EBTA 算法虽然在任务分派的计算时间少占有优势，但当某一地区出现多个事件时，无法与其他执行器节点分担执行任务，只能依次执行，在任务执行时间上大大增加，一旦算例任务数变大，总的任务完成时间也会大大超过本 PSO 算法。

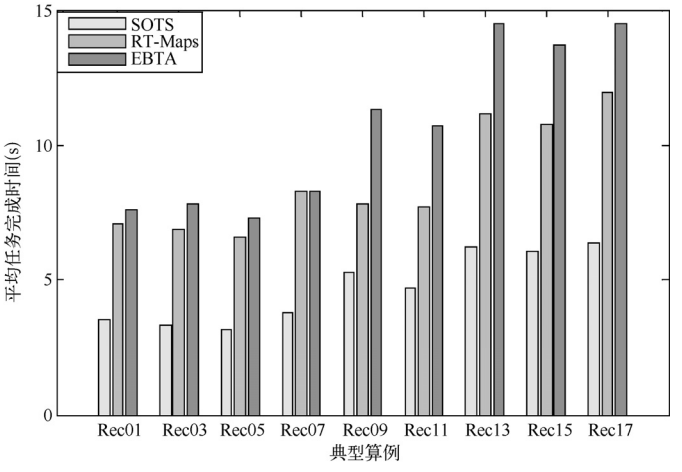


图 8.5　平均任务完成时间比较图

Fig.8.5　Average completion time of task comparison

为了比较算法能耗均衡问题，本书构造了能量均衡指数：

$$C=\frac{\sum_{i=1}^{m}\left(E_{i-\mathrm{all}}/E_{i-\mathrm{rest}}\right)}{m}$$

表示各执行器所消耗能量与剩余能量的平均比值，值越小，表示剩余能量多的执行器承担执行任务多，达到能量均衡的目的。

从图 8.6 观察到，基于 PSO 的任务分派算法的能量均衡指数明显低于 RT-Maps 算法和 EBTA 算法，说明各执行器节点在任务分派算法的调控下，执行任务能耗比较平均，最适合解决同时发生多个复杂事件的场合。而 RT-Maps 算法和 EBTA 算法明显暴露出事件频发时，热点地区的节点能耗较大，整体网络能耗不均衡的问题。

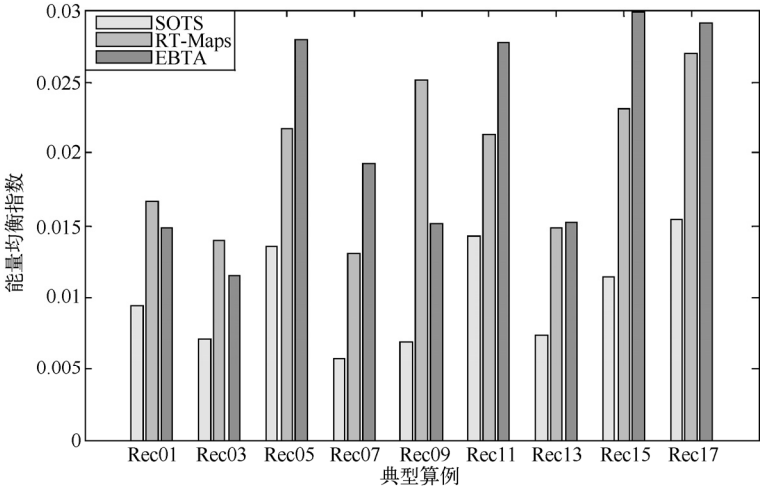


图 8.6　能耗均衡指数比较图

Fig.8.6　Comparison of energy balance index

8.4　小结

随着大规模的执行器节点被整合到网络中，不仅可以对多种传感器数据进行监测、分析处理，同时还可以根据事件类型，做出更快的反应、执行更复杂的逻辑关联性的任务，将成为无线传感器/执行器网络未来的趋势。本章针对多个有执行顺序限制条件的任务同时在多个执行器节点执行的任务分派问题，以任务最大完成时间为优化目标函数，通过执行器节点角色确定降低问题复杂程度，利用微粒群算法优化技术简单易实现、可调参数少、优化速度快等特点，对目标函数进行迭代优化，进行全局搜索。同时，为了进一步优化最优解，按一定概率对已搜索解进行 NEH

方法的局部搜索，确保解空间的充分搜索。仿真、分析结果表明，SOTS 算法搜索速度快，收敛性好，适合解决同时发生多个复杂事件的场合。

参 考 文 献

[1] Howard Hassig Randy Clark et al. SCADA system makes CSO incidents a thing ofthepast.[EB/OL]http://ww.pennnet.com/display_article/308489/41/ARTCL/none/none/1/SCADA-System-Makes-CSO-Incidents-a-Thing-of-the-Past/.

[2] Rappaport. T. Wireless Communication Principles and Practice(2nd Edition)(M). Upper Saddle River, NJ. : London: Prentice Hall PTR. 2002.

[3] J. Kennedy R. C. Eberhart. Particle swarm optimization[C]. Proceedings of the IEEE Int. Conf. Neural Networks IV. 1995:1942-1948.

[4] C. Bean J. Genetic algorithm and random keys for sequencing and optimization[J]. ORSA Journal of Computing. 1994. 6(2):154-160.

[5] Wang L Zheng D. Z. An effective hybrid heuristic for flow shop scheduling problems[J]. Int.J.Adv.Manuf.Technol. 2003. 21: 38-44.

[6] R Reeves C. A genetic algorithm for flowshop sequencing[J]. Computers & Operations Research Oper.Res. 1995,22(1):5-13.

第9章　基于AA协作的多目标任务分派算法

无线传感器/执行器网络的AA协作，主要解决任务分派问题。但是对于多个有工序限制（Ordered Execution）任务在多执行器节点上的分派，目前协作算法在这方面很少研究。第八章针对任务在各执行器节点的协作问题进行探索，提出一种任务调度策略，根据执行器节点的剩余能量和工作状态，利用微粒群（Particle Swarm Optimization，PSO）算法，在任务时效期内，安排各任务在执行器节点上的执行周期，通过构建最小化最大完成时间目标函数，并以节点剩余能量作为约束条件，得到最优任务调度方案。

虽然实时性在无线传感器/执行器网络中是一个非常重要的性能指标，但是，对于网络性能的衡量，不同的应用侧重点也各不相同，单一的优化目标无法贴近真实的应用需求。除了任务完成时间，能量均衡性、数据空间有效性等同样具有优化价值，甚至间接对实时性产生影响。

本章针对无线传感器/执行器网络的AA协作，在第8章分析的基础上，提出一种基于 AA 协作的多目标任务分派算法（Multi-Objective Task Scheduling，MOTS）。算法首先建立多目标任务分派模型，提出任务完成时间、能量均衡指标、存储成本三个优化指标；然后对其已经统一量纲处理，并转化为单目标优化问题求解；利用 PSO 算法全局搜索能力强的特点，同时基于自适应 meta-Lamarchian 学习策略的多邻域模拟退火局部搜索，进一步改进部分解的位置和性能。

9.1　多目标优化问题的基本概念

优化是指在可行解集中搜索目标最优解的过程。如果只有一个优化目标，称为单目标优化问题；如果优化目标在两个或者以上的，称为多目标优化问题。多目标优化（也称多准则、多属性、多指标优化）[1]的定义为：满足约束条件的决策变量使目标函数最优，即寻找决策变量 $\boldsymbol{X}^* = (x_1^*, x_2^*, \ldots, x_n^*)^{\mathrm{T}}$（可能有多个），使目标向量函数 $f(x)$ 最优。其数学描述如下：

$$\begin{cases} \min & f(x)=(f_1(x),f_2(x),\dots f_m(x))^T \\ \text{s.t.} & g_i(x)\leqslant 0, i=1,2,\dots,q \\ & h_j(x)=0, j=1,2,\dots,p \end{cases} \tag{9.1}$$

式中，$\boldsymbol{x}=(x_1,x_2,\dots,x_n)\in \boldsymbol{X}\subset R^n$ 为 n 维的决策矢量；$\boldsymbol{X}$ 为 n 维的决策空间；$f(x)$ 为 m 维的目标矢量；$g_i(\boldsymbol{x})\leqslant 0(i=1,2,\dots,q)$ 定义了 q 个不等式约束；$h_j(\boldsymbol{x})=0(j=1,2,\dots,p)$ 定义了 p 个等式约束。在此基础上，介绍几个重要的定义：

【定义 9.1】对于一个 $x\in \boldsymbol{X}$，如果满足式（9.1）的约束条件，则称 x 为可行解。

【定义 9.2】由 $\boldsymbol{X}$ 中的所有可行解组成的集合，称为可行解集合，记为 $\boldsymbol{X}_f$，且 $\boldsymbol{X}_f\subseteq \boldsymbol{X}$。

【定义 9.3】考虑两个解 x_1 和 x_2，若满足

$$\begin{cases} \forall j\in\{1,2,\dots,m\}, f_j(x_1)\leqslant f_j(x_2) \\ \exists k\in\{1,2,\dots,m\}, f_k(x_1)<f_k(x_2) \end{cases} \tag{9.2}$$

则称解 x_1 支配 x_2，记作 $x_1\succ x_2$。

【定义 9.4】给定一个解 x^*，若在解空间 $\boldsymbol{X}$ 中不存在支配 x^* 的解，则称为 Pareto 最优解，或非支配解，或非劣解（non-dominated solution），即

$$\neg\exists x\in \boldsymbol{X}_f : x\succ x^* \tag{9.3}$$

【定义 9.5】Pareto 最优解集是所有 Pareto 最优解的集合，定义为：

$$P^*\triangleq\{x^*\mid\neg\exists x\in \boldsymbol{X}_f : x\succ x^*\} \tag{9.4}$$

【定义 9.6】Pareto 最优解集 P^* 中所有解对应的目标矢量组成的曲面称为 Pareto 边界（Pareto Front）PF^*：

$$\mathrm{PF}^*\triangleq\{f(x^*)=(f_1(x^*),f_2(x^*),\dots,f_m(x^*))^{\mathrm{T}}\,\big|\,x^*\in P^*\} \tag{9.5}$$

【定义 9.7】设 $f:\Omega\to\Gamma$ 是一个从集合 Ω 到偏序集 $\Gamma(\Gamma,\prec)$ 的一个映射，那么对于 $A\in\Omega$，定义集合 $M_f(A,\prec)=\{a\in A\,|\,f(a)\in M(f(\mathrm{A}),\prec)\}$ 包含了 A 中某些元素，这些元素对应的像在像空间 $f(A)=\{f(a)\,|\,a\in A\}$ 中是极小元。

【定义 9.8】如果 S 是一个有限集，并且 $\{X_t : t\in N_0\}$ 是一个从 S 中取值的随机序列，对于所有的 $t\geqslant 0$ 和所有的 $(i,j)\in S\times S$ 有以下性质：

$P\{X_{t+1}=j\,|\,X_t=i,X_{t-1}=i_{t-1},\dots,X_0=i_0\}=P\{X_{t+1}=j\,|\,X_t=i\}=P_{i,j}$，则称序列

$\{X_t : t \in N_0\}$是一个在状态空间 S 上的齐次有限马尔可夫链。

【定义 9.9】如果一个非负矩阵的每一行元素之和等于 1，则称它是随机的。如有：$\forall i, j \in S : \exists k \in N : P_{ij}(k) > 0$，称随机矩阵 $\boldsymbol{P}$ 是不可约的。

【定理 9.1】一个具有有限状态空间与不可约转移矩阵的齐次马尔可夫链，以概率 1 无限次地访问每一状态而与它的初始分布无关。

9.2 多目标任务分派

本书将完成一个特定目标的行为定义为一个任务，包括通信、数据处理、执行动作等行为。

【定义 9.10】一个由单个执行器节点完成的任务，称为单执行器任务（Single-Actor Task，SAT）；一个任务由多个执行器节点共同完成并具有任务约束条件，则称为多执行器任务（Multi-Actor Task，MAT）。

【定义 9.11】完成一个特定目标的操作（包括通信、数据处理、执行动作等），执行时间和能耗可预知，称为一个任务执行单元，简称任务元，记为 t。

【定义 9.12】一个多执行器任务可分为多个任务执行单元，每个任务元表示在一个执行器节点上的一次执行，一个任务 $T = t_1 \cup t_2 \cup \ldots \cup t_n$，且 $\forall t_i, t_j \in T$，$\exists t_i \cap t_j = \phi$，称任务 T 具有可分性。

【定义 9.13】若 $O = \{o_1, o_2, \ldots, o_n\}$ 为任务 T 的 n 个任务元执行顺序，o_1 只有一个后继任务元，o_n 只有一个前趋任务元，其余任务元拥有一个前趋和一个后继，称其执行具有工序限制（Oredered Execution，OE）。

考虑 n 个任务在 m 个执行器节点上的执行过程，每个任务在各执行器节点上的完成时间已知，并且每个任务必须按照事先规定的顺序完成其在各执行器节点上执行，即任务约束条件。如果一个任务只需一个执行器节点完成，则该任务在其他执行器节点的执行时间用零表示。要求确定在每个执行器节点上执行的所有任务的顺序或开工与完工时间，使得在符合各任务的约束条件的前提下最优化最大完成时间。这里假设：一个执行器节点同时只能执行一个操作；任务没有抢先执行的特权；每一项操作的执行时间和能耗可以估算；操作结果的数据报文格式一致，即传输结果的通信能耗一致。

n 个具有工序限制的任务在 m 个执行器节点上的执行过程，每个任务的执行顺序已知。要让所有任务在一个最短的时间完成，则必须减少最大任务完成时间 $T_{\max}$，以达到实时性要求；同时在安排分派方案的过程中，需要考虑能耗均衡指标 E，尽量延长网络寿命；由于执行器节点存储资源有限，加快数据流转速度，缩短存储时间，降低存储成本 LT，同样对执行

的实时性有积极意义。

9.2.1　最大任务完成时间

参见 8.1.1 节第 7 段开始的内容。

9.2.2　能耗均衡指标

设 $P(t)$ 为单位时间内执行器节点执行任务元所需能耗，γ 表示采集单位长度数据所需能耗系数，则执行器节点 k 所需执行能耗：

$$E_{k\text{-action}}=\sum_{j=1}^{n}\sum_{i=1}^{m}(P(t)P_{j_i,k}+\gamma l)x_{j_i,k} \tag{9.6}$$

可行分配方案必须考虑到执行器节点剩余能量的约束，每个执行器节点的剩余能量不但要足以完成所有任务元序列，还要将执行结果数据传给下一执行器节点或 SINK。根据文献[91]提出的发射硬件能耗模型，r 为执行器节点通信距离，l 表示帧长，则传输和接收能耗公式为

$$\begin{cases}E_{tx}(r,l)=(\alpha r^{n}+\beta)l\\ E_{rx}(l)=\beta l\end{cases} \tag{9.7}$$

式中，ar^n 表示在距离 r 传输的发射功率；β 是发射电路（如 PLLs、VCOs）能耗系数；n 为信道衰减倍数，取决于环境。则一个执行器节点 k 所有任务元所需的总能耗：

$$E_{k\text{-all}}=\sum_{j=1}^{n}\sum_{i=1}^{m}(P(t)p_{j_i,k}+\gamma l+(ar^{n}+\beta)l+\beta l)x_{j_i,k} \tag{9.8}$$

为了比较算法能耗均衡问题，本书构造了能量均衡指数：执行器节点 k 所消耗能量 $E_{k\text{-all}}$ 与剩余能量 $E_{k\text{-rest}}$ 的比值 $E=E_{k\text{-all}}/E_{k\text{-rest}}$，则全网能量均衡目标函数为

$$\min f_2=\min\left\{\sum_{k=1}^{m}\left(E_{k\text{-all}}/E_{k\text{-rest}}\right)\right\} \tag{9.9}$$

比值越小，表示剩余能量多的执行器节点承担执行任务多；反之越小。

9.2.3　存储成本

存储成本是指执行任务元时需要存储的数据长度 L 与开始等待到开始操作这段时间长度 T 的乘积。

设 ST_{j_i} 表示执行任务 j 的第 i 任务元的开始时间，令 $ST_{j_0}=0$，则执行任务元需要等待的时间为

$$T_{j_i}=ST_{j_i}-(ST_{j_{i-1}}+\sum_{k=1}^{m}p_{j_{i-1},k}x_{j_{i-1},k}) \tag{9.10}$$

设 L_{j_i} 表示执行任务 j 的第 i 任务元时的存储数据长度，则总存储成本的目标函数如下：

$$\min f_3=\min\left\{\sum_{j=1}^{n}\left\{\sum_{i=1}^{m}L_{j_i}\left(ST_{j_i}-(ST_{j_{i-1}}+\sum_{k=1}^{m}p_{j_{i-1},k}x_{j_{i-1},k})\right)\right\}\right\} \tag{9.11}$$

9.3 面向 AA 协作的多目标任务分派算法（MOTS）

9.3.1 多目标规范化处理

需要注意的是，多目标优化问题的求解中，绝对的最优解不一定存在，但有效解总是存在的。本书通过理想点法评价函数，将多目标优化问题转换为单目标优化问题，先分别求出问题中各目标函数的最优值，然后让每个目标尽量接近各自的最优值以获得多目标问题的最优解。

设目标 $f=(f_1,f_2,f_3)$，多目标优化模型的评价函数 $u(f)$ 为目标 f 与理想点 f^* 之间的欧式距离：

$$u(f)=\| f-f^* \| \tag{9.12}$$

则可行域内目标 $f=(f_1,f_2,f_3)$ 与理想点 $f^*=(f_1^*,f_2^*,f_3^*)$ 之间的欧式距离最小的解，就是问题的最优解。

9.3.2 随机权值确定

算法的优化过程主要包括搜索解和评价解两个主要环节。在解搜索方面，基于 PSO 算法的搜索方式与求解单目标完全相同。在解评价方面，本章的多目标算法采用随机加权线性累加函数将多个目标整合为一个综合目标来对微粒进行评价。在多目标优化求解过程中，希望从多个方向搜索，获取尽可能多的非劣解，随机线性加权策略[2]展示出良好的有效性。因此，权值按如下方式确定：

$$w_i=\mathrm{random}_i\Big/\sum_{j=1}^{N}\mathrm{random}_j \tag{9.13}$$

式中， random_i 为 0 到 1 之间均匀分布的一个随机数。

考虑到三个不同参数的量纲对目标函数优化的影响，对优化参数做归一化处理，处理后的评价函数为

$$\min u(f)=\sqrt{w_1\left(\frac{f_1-f_1^*}{f_1^*}\right)+w_2\left(\frac{f_2-f_2^*}{f_2^*}\right)+w_3\left(\frac{f_3-f_3^*}{f_3^*}\right)^2\sqrt{a^2+b^2}} \qquad (9.14)$$

了解评价方面的过程如下：

/*解的评价*/

步骤 1：求理想点。单独求出最大完成时间函数的最小值 f_1^* 及最优解 x_1；能耗均衡指标的最小值 f_2^* 及最优解 x_2；存储成本的最小值 f_3^* 及最优解 x_3，得到理想点。可用 PSO 算法获得。

步骤 2：检验理想点。检验最优解 x_1、x_2、x_3 是否相等，如果相等，则得到绝对最优解 $x^*=x_1=x_2=x_3$，算法结束；否则，转步骤 3。

步骤 3：随机产生权值 w_1、w_2、w_3，求解多目标优化问题，解出有效解 $\tilde{x}$。

步骤 4：如果算法终止条件达到，则由 $\tilde{x}$ 得到任务分派方案，算法结束；否则转入步骤 3。

9.3.3　执行器节点角色确定

参见 8.2.1 节。

9.3.4　标准微粒群优化算法

微粒群优化算法（Particle Swarm Optimization，PSO）[3]是一种进化计算技术，由 Eberhart 博士和 Kennedy 博士发明。与其他进化类算法类似，PSO 也采用“群体”与“进化”的概念，同样也是依据个体（粒子）的适应值大小进行操作。不同之处在于 PSO 算法并不对个体使用进化算子，而是将每个个体看成在 d 维搜索空间中的一个没有质量和体积的微粒，并在搜索空间中以一定的速度飞行。基本原理是：PSO 首先在可行解空间和速度空间随机确定微粒的初始位置和初始速度，其中位置用于表征问题解。

$X_i=[x_{i,1},x_{i,2},\ldots,x_{i,d}]$ 为 d 维搜索空间的第 i 个微粒的位置；

$V_i=[v_{i,1},v_{i,2},\ldots,v_{i,d}]$ 为 d 维搜索空间的第 i 个微粒的速度；

$P_i=[p_{i,1},p_{i,2},\ldots,p_{i,d}]$ 为 t 时刻 d 维搜索空间的第 i 个微粒的最佳位置，由微粒的适应性函数 $\text{fitness}(X)$ 决定，函数值越小代表微粒适应性越好。微粒 i 的最佳位置由式（9.22）确定：

$$p_i(t)=\begin{cases}p_i(t-1) & if \quad \text{fitness}(x_i(t))\geqslant \text{fitness}(p_i(t-1))\\ x_i(t) & if \quad \text{fitness}(x_i(t))\leqslant \text{fitness}(p_i(t-1))\end{cases} \tag{9.15}$$

式中，t 表示迭代次数。再根据每个微粒所经过的最佳位置（P_{best}）确定群体最佳位置 $p_{\text{g}}(t)$，并按如下公式分别更新各微粒的速度和位置：

$$v_{i,j}(t+1)=wv_{i,j}(t)+c_1r_1[p_{i,j}(t)-x_{i,j}(t)]+c_2r_2[p_{\text{g},j}(t)-x_{i,j}(t)] \tag{9.16}$$

$$x_{i,j}(t+1)=x_{i,j}(t)+v_{i,j}(t+1),j=1,...,d \tag{9.17}$$

式中，w 为惯性权因子；c_1 和 c_2 为正的加速常数；r_1 和 r_2 为在 0 到 1 之间均匀分布的随机数。

9.3.5 基于 ROV 规则的编码

由于 PSO 算法中微粒的位置为连续值矢量，无法表示任务的执行排序，因此首先必须对微粒位置矢量进行编码，构造其与任务排序的恰当映射，然后才能利用标准 PSO 算法解决任务分派问题。

微粒的位置矢量 $\boldsymbol{X}_i=[x_{i,1},x_{i,2},...,x_{i,n}]$ 可以由随机键编码[4]产生，即在一定范围内随机产生一些实数。但是这些用实数表示的位置分量本身无法表示任务的执行顺序，因此必须进行转换。为了保证编码策略不遗漏问题的全局最优解，并适应标准 PSO 算法，引入一种针对随机键编码的基于升序排列（Ranked Order Value，ROV）规则，实现从微粒的连续位置矢量到任务排序的转化，使之适应问题求解。基于随机键编码产生的位置分量虽然是一些实数，但是分量之间存在大小差异，利用这个差异可将连续的位置矢量转化为离散的执行排序，具体规则如下：

（1）找出 $x_{i,1},x_{i,2},...,x_{i,n}$ 中值最小的分量 $x_{i,d}$，并令对应的加工排序值 j_d 的 ROV 值为 1。

（2）找出除开 $x_{i,d}$ 的值最小的分量，并令其对应的 ROV 值为 2；重复直到找完所有位置分量。

此时，微粒的位置矢量 $\boldsymbol{X}_i=[x_{i,1},x_{i,2},...,x_{i,n}]$ 转换为离散的执行排序 $\pi=(j_1,j_2,...,j_n)$。需要注意两点：（1）如果位置矢量中出现多个相同值的分量，需要随着位置的增加依次累加一个足够小的正数，使得这些位置的分量值互不相同；（2）对于局部搜索操作，如互换操作 SWAP、逆序操作 INVERSE 和插入操作 INSERT[5]，位置分量与对应的 ROV 值是一一对应的，排序发生变化，位置分量相应的也发生变化，调整非常简单。

9.3.6　多目标微粒群搜索

多目标微粒群算法在搜索解空间方面与单目标算法相同，不同之处在于，处理多目标整合时采取了随机加权，种群中的每一个微粒都由式 9.13 产生一组权值。因为权值各不相同，则在进化过程中，同时有 p_s 个微粒并行地在多个方向上进行搜索。然而，在一次进化过程中，每个微粒的随机权值是不变的，特别需要注意在更新微粒的最佳位置时，新旧位置的评价所用到的随机权是一致的。为了保存非劣解，算法设置了非劣解集。一旦算法得到新解，首先比较新解与 pbest 的支配关系。若存在支配关系，则将其中的非支配解作为新的 pbest；若不存在支配关系，则用该微粒的随机权带入目标函数进行评价，选择较好的解作为 pbest。直到所有的微粒最佳位置都得到更新，算法再产生一组新的随机权对所有 pbest 和群体最佳位置 gbest 进行比较，选出最好的解作为 gbest。

9.3.7　基于自适应学习策略的多目标局部搜索

微粒群算法原理简单，容易实现，但由于收敛速度快，不能够保证绝对搜索到全局最优解，容易陷入局部极小解。本章的优化对象主要是任务的排列，研究表明增加对邻域结构的局部搜索，有助于避免算法早熟收敛。在此引入三种邻域结构。

SAWP：随机交换任务执行顺序中的两个不同任务的位置。

INSERT：随机选择一任务，将其插入到任务执行顺序中的另一个随机位置。

INVERSE：将任务执行顺序中两个不同的随机位置间的任务逆序。

充分利用这三种邻域操作，扩大搜索空间，对找到更好的非劣解是有帮助的。但是如果选择了不合适的邻域，同样对算法搜索造成负面影响。Ong 和 Keane[6]提出了求解连续优化问题的一种 meta-Lamarckian 学习策略。通过在算法搜索阶段的学习，自适应地从多个邻域结构中选择最适合的一个结构，进行局部搜索。需要注意的是：搜索对象不但包括群体最佳位置 gbest 对应的排列，为了搜索尽可能多的非劣解，还对所有微粒的最佳位置 pbest 对应的排列施行按一定概率的多邻域搜索。

基于自适应学习策略的多目标局部搜索分为两个阶段：训练阶段（Training Phase）和非训练阶段（Non-Training Phase）。

训练阶段的主要目标是采用三种不同的邻域结构，对微粒最佳位置和群体

最佳位置进行基于模拟退火的局部搜索，算出每种邻域结构的奖励值，以此为基础，便于算法在非训练阶段计算每种邻域的选择概率。训练阶段只在首次执行局部搜索时实施，以后的搜索过程都属于非训练阶段。

假设局部搜索前的 gbest 对应目标值为 fitness(gbest)，而微粒 i 的 pbest_i 对应的目标值为 $\text{fitness}(\text{pbest}_i)$，利用模拟算法进行 $n(n-1)$步搜索，找出在第 k 种邻域下的群体最佳目标值 $\text{fitness}(\text{gbest},k)'$ 和各微粒最佳目标值 $\text{fitness}(\text{pbest}_i,k)'$，并按下列公式计算出第 k 种邻域结构的奖励值 $\eta(\text{gbest},k)$ 和 $\eta(\text{pbest}_i,k)$：

$$\eta(\text{gbest},k)=\frac{\text{fitness}(\text{gbest})-\text{fitness}(\text{gbest},k)'}{n(n-1)} \tag{9.18}$$

$$\eta(\text{pbest}_i,k)=\frac{\text{fitness}(\text{pbest}_i,k)-\text{fitness}(\text{pbest}_i,k)'}{n(n-1)} \tag{9.19}$$

/*训练阶段*/

1．find $\text{fitness}(\text{gbest},\text{SWAP})'$, $\text{fitness}(\text{gbest},\text{INSERT})'$, $\text{fitness}(\text{gbest},\text{INVERSE})'$ by using SA

2．compute $\eta(\text{gbest},\text{SWAP})$, $\eta(\text{gbest},\text{INSERT})$, $\eta(\text{gbest},\text{INVERSE})$ from equation 9.25

3．for (i = 1; i > ps; i++)　　//ps 为微粒群规模

4．find $\text{fitness}(\text{pbest}_i,\text{SWAP})'$, $\text{fitness}(\text{pbest}_i,\text{INSERT})'$, $\text{fitness}(\text{pbest}_i,\text{INVERSE})'$ by using SA

5．compute $\eta(\text{pbest}_i,\text{SWAP})$, $\eta(\text{pbest}_i,\text{INSERT})$, $\eta(\text{pbest}_i,\text{INVERSE})$ from equation 9.19

对于在训练阶段，每种邻域执行进行 $n(n-1)$步搜索的模拟退火算法：

/*模拟算法伪代码*/

1．set $i=0$　//抽样初值

2．while $T_{\text{cur}}>T_{\min}$ and $i<n(n-1)$ do　//n 为任务数

3．for ($k=0$; $k<3$; $k++$)　//三种邻域操作

4．compute $\text{fitness}(x,k)$; $\Delta E(k)=\text{fitness}(x,k)-\text{fitness}(x_{\text{best}},k)$

5．if $\Delta E(k)<0$　$x_{\text{best}}(k)=x(k)$

6．else

7．{　$p=\exp(-\Delta E(k)/T_i)$

8．if $\text{random}[0,1]<p$　$x_{\text{best}}(k)=x(k)$

9．else $x_{\text{best}}(k)=x_{\text{best}}(k)$　}

10．$n++$

11．output　$x_{\text{best}}(\text{SWAP})$; $x_{\text{best}}(\text{INSERT})$; $x_{\text{best}}(\text{INVERSE})$

一旦在初温时得到群体最佳位置和各微粒最佳位置在每种邻域下的奖励值以后，此后的局部搜索就进入非训练阶段。在此阶段，根据奖励值计算每种邻域的选择概率：

$$P_{\text{neigh}}\left(\text{pbest}_i,k\right)=\frac{\eta(\text{pbest}_i,k)}{\sum_{k=1}^{3}\eta(\text{pbest}_i,k)} \tag{9.20}$$

并利用轮盘赌策略选定一个结构，在该邻域下进行基于模拟退火的局部搜索。当局部搜索完毕后，需要更新奖励值 $\Delta\eta(\text{pbest}_i,k)$，由式（9.19）计算出，同时更新微粒最佳位置 pbest 和群体最佳位置 gbest。

$$\eta(\text{pbest}_i,k)=\eta(\text{pbest}_i,k)+\Delta\eta(\text{pbest}_i,k) \tag{9.21}$$

/*非训练阶段*/
1．compute　$P_{\text{neigh}}(\text{gbest},k)$　and　$P_{\text{neigh}}(\text{pbest}_i,k)$
2．chance one neighborhood struct for *gbest* and *pbest* of each particle
3．run SA at this struct
4．update T and　η　　//更新温度和奖励值
5．update *gbest* and *pbest* of each particle　　//更新群体最佳位置和微粒最佳位置

通过采用自适应学习策略，多种邻域结构按贡献选择使用，不但丰富了搜索方式，而且可以避免不合适的邻域对算法搜索带来的负面影响。同时在选中的邻域下进行模拟退火算法的局部搜索，进一步改善解的质量，考虑到计算量，利用退温速率进行控制。

9.3.8　MOTS 算法流程与分析

MOTS 算法首先对网络执行器节点的能量等情况进行评估，确定各自角色，便于降低任务分派的排列复杂度；然后利用 PSO 算法收敛速度快，搜索能力强的特点进行全局搜索，利用 ROV 编码进行目标函数评估。需要注意的是：由于是随机加权的多目标函数，每个微粒都会产生各自的一组随机权值，并在这一代进化过程中，评价新旧解以及局部搜索中保持不变，直到该过程完毕，下一代进化时再重新产生新的随机权值；对于局部搜索，算法采用多邻域搜索，为了找到合适的邻域结构，利用了 meta-Lamarckian 学习策略，使得算法使用合适的结构进行模拟退火算法的局部搜索，进一步改善对应的任务排列。图 9.1 为 MOTS 算法流程。

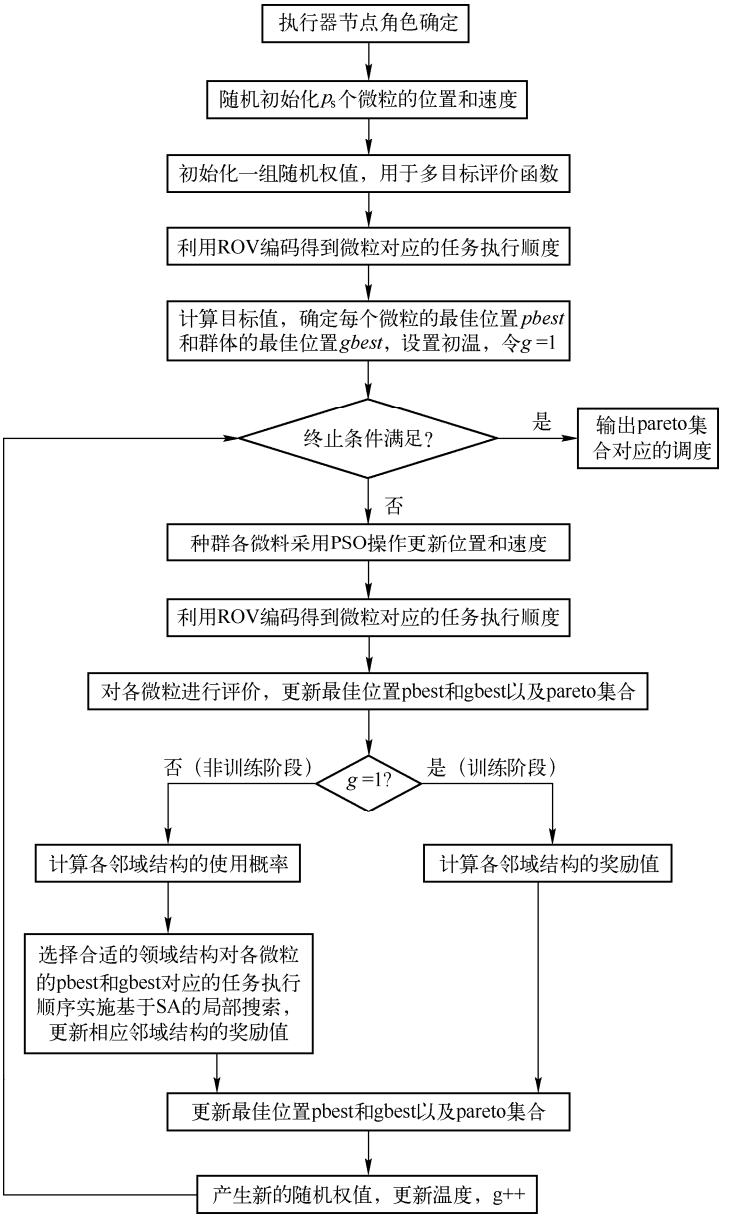


图 9.1　MOTS 算法流程图

Fig.9.1　Flowchart of MOTS

MOTS 算法实现：

/*MOTS 算法伪代码*/

1．random initialize　$x_i(0)$，$1 \leqslant i \leqslant p_s$　// p_s 为群体规模

2．crease a group random weight for each particle

3．compute single-object idea point of fitness( )
4．get task sequence for each partical by ROV coding
5．compute fitness($x_i(0)$) for each partical from equation 5.6
6．random initialize $v_i(0)$
7．set $\text{pbest}_i = x_i(0)$, and gbest = min(fitness($x_i(0)$))
8．set noChangeGen = 0 and g = 1 and T
9．while noChangeGen < L do　//重复执行下列步骤，直到 pareto set 目标值连续 L 步保持不变
10．update $x_i(t)$, $v_i(t)$
11．get task sequence for each partical by ROV coding
12．compute fitness($x_i(t)$) for each partical
13．update pbest_i and gbest and pareto set
14．if (g == 1) goto traning phase //如果是首次执行局部搜索，进入训练阶段
15．else goto training phase　//否则，进入非训练阶段
16．update pbest_i and gbest and pareto set
17．if (new gbest ==old gbest), noChangeGen = noChangeGen + 1
18．else noChangeGen = 0
19．crease a group new random weight for each particle
20．update T and g++
21．output pareto set and corresponding task sequence of pareto set

【定义 9.14】令 X_t 为算法在第 t 代的微粒群，$F_t = f(X_t)$ 是相应的适应函数值集合，F^* 是任务调度的搜索空间 Ω 内的极小元集，则如果 $\lim\limits_{t\to\infty}\zeta_{F^*}(F_t)\lim\limits_{t\to\infty}\{|F_t|-|F_t\cap F^*|\}\to 0$ 以概率 1 成立，则称算法以概率 1 收敛于极小元。

【定理 9.2】MOTS 算法的微粒群搜索过程是具有正转移矩阵的齐次有限马尔可夫链。

【证明】虽然算法求解的是一个连续问题，但在确定的精度意义下，仍假设搜索空间是有限的。群体的每个微粒的当前位置为一个状态，群体下一代所处状态的概率仅取决于当前状态，而与以前状态无关，即群体无后效性。同时各代微粒群之间的转移矩阵与时间的起点无关，即

$$P\{X_{t+1}=j\,|\,X_t=i,X_{t-1}=i_{t-1},\ldots,X_0=i_0\}=P\{X_{t+1}=j\,|\,X_t=i\}=P_{i,j}$$

根据有限齐次马尔可夫链的定义 9.8 可知，MOTS 算法的微粒群序列的转移矩阵 $\boldsymbol{P}$ 是与时间 t 无关的有限齐次马尔可夫链。

现在证明转移矩阵是正的。MOTS 算法中微粒群序列的转移矩阵主要涉及微粒群的更新式（9.23）和（9.24），因速度更新公式中存在的 r_1、r_2 以及 $p_{i,j}$ 和 $p_{g,j}$ 在 Pareto 解集中的随机性，使得 $P(v_{i,j}(t)\neq 0)>0$，通过迭代进化，每个微粒在下一代存在变化成任意一个个体（解空间中任意位置）的可能性，因此它

的转移矩阵是正的。

对于算法中的局部搜索，三种多邻域结构都可以通过操作，存在演化成任意一个个体的可能性，因而在局部搜索阶段，转移矩阵仍然是正的。

综上所述，MOTS 算法中的微粒群序列 $(A_t)_{t\geqslant 0}$ 是具有正转移矩阵的齐次有限马尔可夫链。

【定理 9.3】如果 MOTS 算法的微粒群序列 $(A_t)_{t\geqslant 0}$ 是具有正转移矩阵的齐次有限马尔可夫链，那么该算法以概率 1 收敛于极小元集合。

【证明】设 $f(A_t)$ 是 A_t 的适应值向量集，$\Gamma^* = M(\Gamma,\leqslant)$ 是问题的极小元集，且是完全的（即 Pareto 最优解集）。

在 MOTS 算法的执行过程中，偏序符号“$\prec$”的应用其实就是非劣解的求解过程，包括全局搜索和局部搜索中群体最佳位置和各微粒最佳位置的更新，找到的非劣解将进入 Pareto 最优解集 r_{best}。算法迭代的最终结果是 $f(r_{\text{best}}) \to \Gamma^*$，从以下两方面证明。

首先，当微粒找到一个非劣解，且与 Pareto 最优解集 r_{best} 中的某个微粒 a 的比较中占优（具有支配关系），将微粒 a 从 r_{best} 中删除，非支配解进入 r_{best} 中，当且仅当存在一个非劣解 $b \in A_t$ 使得 $f(b) < f(a)$。根据 $\Gamma^* = M(\Gamma,\leqslant)$ 的定义，Γ^* 中的微粒一旦进入 r_{best}，就永远不会被删除。

其次，令微粒 $a \in r_{\text{best}}$ 且 $f(a) \notin \Gamma^*$，根据定义 9.8，一定存在一个微粒 $b \in A_t$ 满足 $f(b) < f(a)$，根据 r_{best} 的更新准则，只要这个更好的微粒能够经过一定的迭代次数 t 在 $(A_t)_{t\geqslant 0}$ 中出现，它一定会进入 r_{best} 取代 a。随着迭代过程不断发生，直到 r_{best} 中没有非最优微粒。由于 $(A_t)_{t\geqslant 0}$ 是具有正转移矩阵的齐次有限马尔可夫链，定理 9.1 保证了解空间中的所有位置都会以概率 1 被无限次访问。因此，r_{best} 中的非最优微粒经过有限次迭代以后会以概率 1 被淘汰。

综上所述，所有进入 r_{best} 的最优解都不会被淘汰，而所有 r_{best} 中的非最优解都会以概率 1 在有限步内被淘汰。因此，当 $T\to\infty$ 时，$f(r_{\text{best}}) \subset \Gamma^*$，有 $\lim\limits_{t\to\infty} \zeta_{F^*}(F_t) \to 0$ 以概率 1 成立，根据定义 9.14，MOTS 算法以概率 1 收敛于极小元。

【定理 9.4】面向 AA 协作的微粒群任务分派算法的空间复杂度为 $\mathrm{O}((5p_s+2)n)$，群体每次进化的最坏渐进时间复杂度为 $\mathrm{O}(mn+p_s+5Lp_sn)$，其中 p_s 为群体规模，n 表示任务数量，m 为执行器节点数量，L 为进化次数。

【证明】在微粒群任务分派算法过程中，每次进化需要存储 p_s 个微粒的当前位置、微粒最佳位置、速度和适应目标函数值，所消耗的存储空间为 $\mathrm{O}(4p_sn)$；多邻域的模拟退火局部搜索时需要计算每个微粒的选择概率，存储空间为 $\mathrm{O}(p_sn)$；邻域搜索时还需要一个存储新位置或速度的空间，加上一个存储群体

最优调度，合为 $O(2n)$。所以算法的空间复杂度为 $O((5p_s+2)n)$。

执行器节点角色确定，时间复杂度为 $O(mn)$，多目标函数的理想点计算时间复杂度为 $O(3n^2)$，初始化 p_s 个微粒的时间复杂度为 $O(p_s)$，每个微粒确定历史最优微粒和全局最优微粒、ROV 编码、计算选择概率和计算适应函数的时间复杂度均为 $O(p_s n)$，多邻域局部搜索为 $n(n-1)$，则一次进化的时间复杂度为 $O(4p_s n+n^2)$，进化过程要执行 L 次，则重复进化过程中的时间复杂度为 $O((4p_s n+n^2)L)$，算法的总时间复杂度为 $O(mn+3n^2+p_s+(4p_s n+n^2)L)$。

9.4　算法仿真与性能分析

9.4.1　实验参数

为了验证算法的有效性，选取了文献[7]中的实验 1（8 个任务 8 个执行器节点，共 27 个任务元）、实验 2（10 个任务 10 个执行器节点，共 30 个任务元）、实验 3（15 个任务 10 个执行器节点，共 56 个任务元）和文献[8]中的实验 4（10 个任务 7 个执行器节点，共 30 个任务元等 4 个轻量级测试算例进行比较）。

任务参数见表 9.1。

表 9.1　任务参数表

Table9.1　Parameters of task

实　验	任 务 数 量	执行器节点数量	分解后任务元数量
1	8	8	27
2	10	10	30
3	15	10	56
4	10	7	30

同时为了适应无线传感器/执行器网络的特点，对四个实验条件进行改造，增加存储条件和能量消耗条件如下：

存储条件：每个任务在各执行器节点执行后，将产生 128Byte 长度的数据。

能耗条件：给出单位时间的执行能耗 5000nJ。

仿真硬件环境：Intel Pentium IV/2.2GHz/512M RAM。

软件平台：Windows XP，MATLAB 7.0。

算法参数见表 9.2。

表 9.2　MOTS 算法参数表

Table9.2　Parameters of MOTS

PSO		SA	
群体规模 p_s	20		
惯性权因子 w	1.0	初温 T_0	3/5/10/5
加速因子 c_1, c_2	2	SA 结束温度 T_{end}	0.01
微粒最小位置值 x_{min}	0	退温速率	0.9
微粒最大位置值 x_{max}	4.0	终止条件参数 L	5
微粒最小速度值 v_{min}	−4.0	温度更新系数 λ	0.95
微粒最大速度值 v_{max}	4.0	抽样步数	$n(n-1)$

9.4.2　算法性能实验

为衡量多目标算法性能，引入一些性能指标用以评价所得非劣解集的质量，并与典型进化算法模拟退火 SA、遗传算法 GA 等其他算法进行比较，见表 9.3。

（1）NNV（the Number of Non-dominated Vector）

该指标定义为非劣解集中的解个数。

（2）GD（Generational Distance）

该指标由文献[9]提出，表示集合中解与理论上最优 pareto 边界之间的距离，定义如下：

$$\mathrm{GD}=\frac{\sum_{i=1}^{N} d_i}{N}$$

对于集合中的解，最优 pareto 边界中总存在一个与其距离最近的解，其中 d_i 表示两者之间的距离。

（3）AQ（Average Quality）

文献[111]中引入一个分散性指示函数，以一系列具有典型权值的尺度函数对非劣解集进行采样求值，反映解集的近似性和分散性的综合性能。公式省略，这里设定 $\rho=0.01$，$r=10$。

（4）RT（Running Time）

该指标表示算法的运行时间，反映算法的优化效率。

表 9.3　算法性能比较

Table 9.3　Performance comparison

指　　标	算　　法	实验 1[7]	实验 2[7]	实验 3[7]	实验 4[8]
NNV	MOTS	2	4	5	5
	SA	3	5	5	6
	GA	3	5	6	6
GD	MOTS	0.012	0.46	0.72	0.107
	SA	4.43	7.87	11.45	3.28
	GA	1.47	5.12	8.43	2.73
AQ	MOTS	867.3	923.1	947.2	877.5
	SA	894.7	976.2	983.7	911.6
	GA	891.2	964.8	978.3	902.4
RT	MOTS	8.3	14.7	145.5	6.5
	SA	34.7	65.2	422.1	27.9
	GA	137.6	245.1	976.2	96.3

与典型的智能优化算法 GA 和 SA 比较，除了算法所得非劣解个数少以外，其余各项性能指标均明显好于对方：GD 表明解集更接近最优 pareto 边界；AQ 反映算法的综合性能更优；RT 说明了 PSO 算法的效率较高。

9.4.3　网络性能实验

网络性能实验主要将本算法与典型算法 RT-Maps、EBTA 进行比较，考察响应时间和能耗均衡两方面的情况，实验安排 10 个执行器节点，随机部署在 100m×100m 的范围内。为集中考察算法在任务分派上的性能，排除通信中断引起的多余能耗，每个节点的通信半径都设为 100m，遵循无线通信协议 IEEE 802.11。其他参数设置见表 9.4。

表 9.4　执行器节点参数表

Table9.4　Parameters of actor

执行器空闲等待能耗	0J	通信距离 r	100m
数据包长 l	128Byte	发送/接收电路能耗 β	$50nJ\cdot bit^{-1}$
放大倍数 α	$0.0013pJ\cdot(bit\cdot m^{-2})^{-1}$	Two-ray ground 模型指数 n	4
采集能耗指数 γ	$5nJ\cdot bit^{-1}$	单位时间内执行能耗 P(t)	5000nJ

10 个执行器节点的初始能量见表 9.5。

表 9.5　执行器节点初始能量

Table9.5　Initial energy of actor

执行器号/初始能量（J）	1/1	2/3	3/1.5	4/2.5	5/2	6/1	7/2	8/2.5	9/2	10/1

由图 9.2 可知，MOTS 算法将任务的最大完成时间作为优化目标之一，通过分解任务元，使得多个任务元在各执行器节点上并行执行，最大程度上缩短了任务最大完成时间。从实验效果上看，MOTS 算法的最大完成时间指标明显好于其他两种算法，尤其是在实验 2 上，MOTS 算法的最大完成时间只占 RT-maps 算法的 30.4%，占 EBTA 算法的 28%，说明如果不进行任务细分，每个任务线性排列执行，尤其在任务频发区域，任务完成时间将会大大延长。

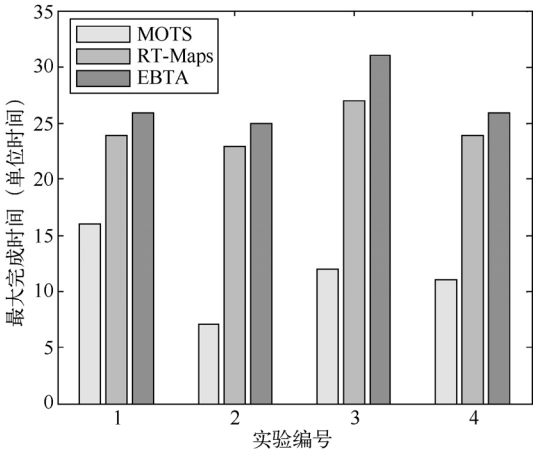


图 9.2　任务最大完成时间对比

Fig.9.2　The largest task completion time comparison

能量均衡指标 E 的方差是用来考察任务执行完毕后，各执行器节点的能量平均分布情况。由图 9.3 可以观察到，由于 MOTS 算法并行安排任务元在各执行器节点上执行，最大程度上平均分担了任务的执行能耗，与其他两种算法相比较，能耗更加均衡，有利于延长整个网络的生存期。

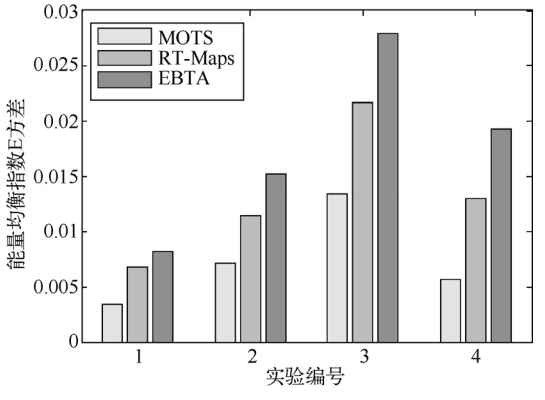


图 9.3　任务能耗均衡指数对比

Fig.9.3　Task energy balance index comparison

从图 9.4 的四个实验可以观察到，由于本算法采用了多目标任务调度，合理安排各任务元在执行器节点上的执行顺序，数据滞留时间变短，其存储成本明显好于其他两种算法。

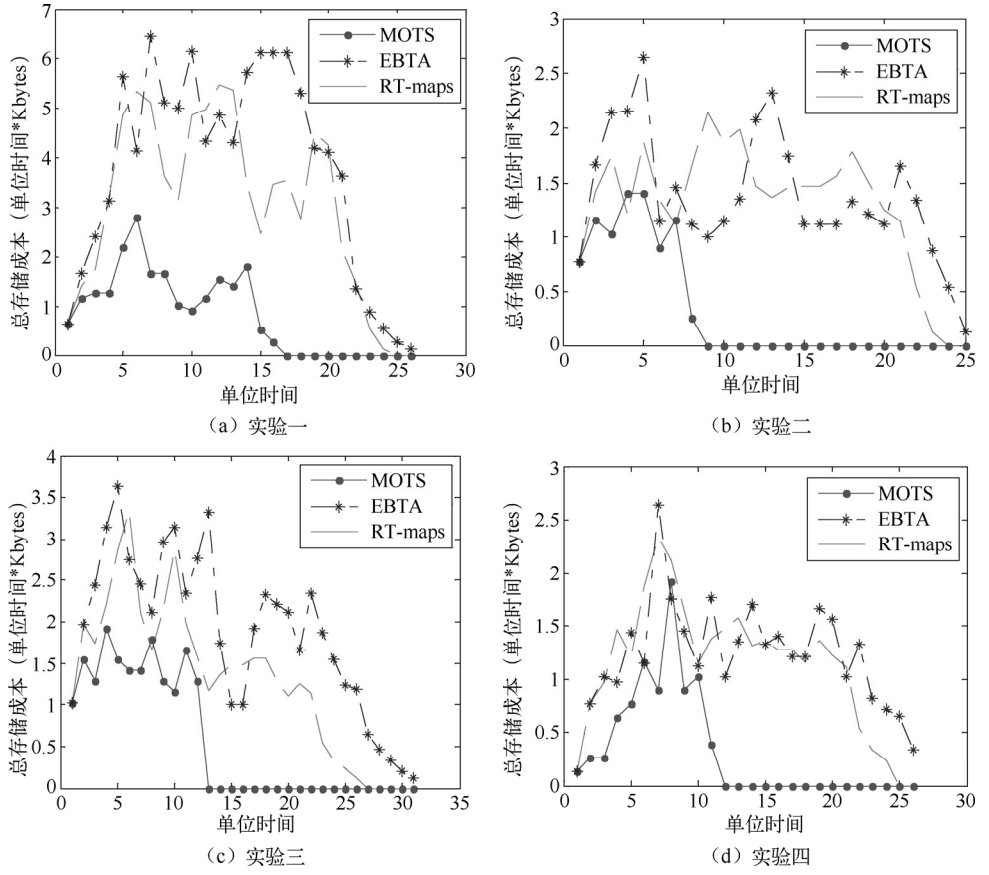


图 9.4　存储成本对比

Fig.9.4　Storage Cost Comparison

9.5　小结

随着大规模的执行器节点被整合到网络中，不仅可以对多种传感器数据进行监测、分析处理，同时还可以根据事件类型，做出更快的反应、执行更复杂的逻辑关联性的任务，将成为无线传感器/执行器网络未来的趋势。本章针对多个有执行顺序限制条件的任务同时在多个执行器节点执行的任务分派问题，以最大完成时间、能耗均衡指标、存储成本为优化目标，利用理想点法将多目标

优化规范化为单目标优化，得到各任务在各执行器节点上的多目标动态调度方案。通过执行器节点角色确定降低问题复杂程度，利用微粒群算法优化技术简单易实现、可调参数少、优化速度快等特点，对目标函数进行迭代优化，进行全局搜索。同时，为了进一步优化最优解，对已搜索解进行基于自适应学习策略的多邻域局部搜索，确保解空间的充分搜索。仿真、分析结果表明，基于 PSO 的任务分派算法搜索速度快，收敛性好，各执行器节点的实时性、能耗等性能指标均优于比较算法，适合解决同时发生多个复杂事件的场合。

参 考 文 献

[1] Deb, K. Pratap. A Agarwal, S. and Meyarivan T. A fast and elitist multi-objective genetic algorithm: NSGA-II[J]. IEEE Transaction on Evolutionary Computation, 2002.6(2):181-197.

[2] Ishibuchi H Murata T. A. multi-objective genetic local search algorithm and its application to flowshop scheduling[J]. IEEE Trans. Syst. Man.Cy.B. 1998, 28 (3):392-402.

[3] J. Kennedy R. C. Eberhart. Particle swarm optimization[C]. Proceedings of the IEEE Int. Conf. Neural Networks IV. 1995:1942-1948.

[4] C. Bean J. Genetic algorithm and random keys for sequencing and optimization[J]. ORSA Journal of Computing. 1994. 6(2):154-160.

[5] Wang L Zheng D. Z. An effective hybrid heuristic for flow shop scheduling problems[J]. Int.J.Adv.Manuf.Technol. 2003. 21: 38-44.

[6] Ong Y S Keane A. J. Meta-Lamarckian learning in memetic algorithms[J]. IEEE Trans. Evolut. Comput. 2004,8(2):99-110.

[7] Xia Weijun, Wu Zhiming. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems[J]. Computers and Industrial Engineering 2005. 48(2): 409-425.

[8] Kacem. Imed Genetic algorithm for the flexible job-shop scheduling problem [C]. Proceedings of the IEEE International Conference on Systems. Man and Cybernetics. 2003.4:3464-3469.

[9] AR Rahimi-Vahed SM Mirghorbani J. Comb. A multi-objective particle swarm for a flow shop scheduling problem[J]. Journal of combinatorial optimization. 2007.13(1):79-102.

反侵权盗版声明